Neural Networks and Rational Functions

Matus Telgarsky [1]

Abstract

Neural networks and rational functions efficiently approximate each other. In more detail, it is shown here that for any ReLU network, there exists a rational function of degree $\mathcal{O}(\text{poly}\log(1/\epsilon))$ which is ϵ-close, and similarly for any rational function there exists a ReLU network of size $\mathcal{O}(\text{poly}\log(1/\epsilon))$ which is ϵ-close. By contrast, polynomials need degree $\Omega(\text{poly}(1/\epsilon))$ to approximate even a single ReLU. When converting a ReLU network to a rational function as above, the hidden constants depend exponentially on the number of layers, which is shown to be tight; in other words, a compositional representation can be beneficial even for rational functions.

1. Overview

Significant effort has been invested in characterizing the functions that can be efficiently approximated by neural networks. The goal of the present work is to characterize neural networks more finely by finding a class of functions which is not only well-approximated by neural networks, but also well-approximates neural networks.

The function class investigated here is the class of *rational functions*: functions represented as the ratio of two polynomials, where the denominator is a strictly positive polynomial. For simplicity, the neural networks are taken to always use ReLU activation $\sigma_r(x) := \max\{0, x\}$; for a review of neural networks and their terminology, the reader is directed to Section 1.4. For the sake of brevity, a network with ReLU activations is simply called a *ReLU network*.

1.1. Main results

The main theorem here states that ReLU networks and rational functions approximate each other well in the sense

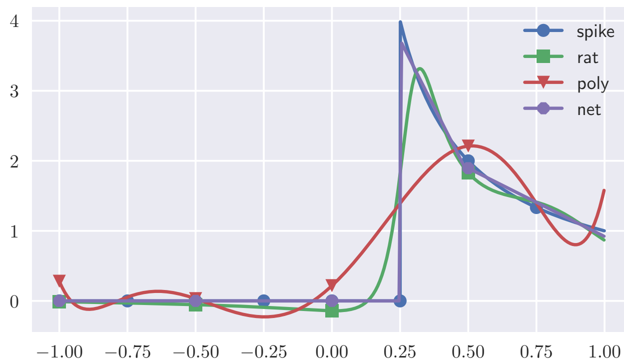

Figure 1. Rational, polynomial, and ReLU network fit to "spike", a function which is $1/x$ along $[1/4, 1]$ and 0 elsewhere.

that ϵ-approximating one class with the other requires a representation whose size is polynomial in $\ln(1\,/\,\epsilon)$, rather than being polynomial in $1/\epsilon$.

Theorem 1.1. *1. Let $\epsilon \in (0,1]$ and nonnegative integer k be given. Let $p : [0,1]^d \to [-1,+1]$ and $q : [0,1]^d \to [2^{-k}, 1]$ be polynomials of degree $\leq r$, each with $\leq s$ monomials. Then there exists a function $f : [0,1]^d \to \mathbb{R}$, representable as a ReLU network of size (number of nodes)*

$$\mathcal{O}\Big(k^7 \ln(1\,/\,\epsilon)^3 + \min\Big\{srk\ln(sr\,/\,\epsilon), sdk^2\ln(dsr\,/\,\epsilon)^2\Big\}\Big),$$

such that

$$\sup_{x\in[0,1]^d}\left|f(x) - \frac{p(x)}{q(x)}\right| \leq \epsilon.$$

2. Let $\epsilon \in (0,1]$ be given. Consider a ReLU network $f : [-1,+1]^d \to \mathbb{R}$ with at most m nodes in each of at most k layers, where each node computes $z \mapsto \sigma_r(a^\top z + b)$ where the pair (a,b) (possibly distinct across nodes) satisfies $\|a\|_1 + |b| \leq 1$. Then there exists a rational function $g : [-1,+1]^d \to \mathbb{R}$ with degree (maximum degree of numerator and denominator)

$$\mathcal{O}\left(\ln(k/\epsilon)^k m^k\right)$$

such that

$$\sup_{x\in[-1,+1]^d}\left|f(x) - g(x)\right| \leq \epsilon.$$

[1]University of Illinois, Urbana-Champaign; work completed while visiting the Simons Institute. Correspondence to: your friend <mjt@illinois.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

Perhaps the main wrinkle is the appearance of m^k when approximating neural networks by rational functions. The following theorem shows that this dependence is tight.

Theorem 1.2. *Let any integer $k \geq 3$ be given. There exists a function $f : \mathbb{R} \to \mathbb{R}$ computed by a ReLU network with $2k$ layers, each with ≤ 2 nodes, such that any rational function $g : \mathbb{R} \to \mathbb{R}$ with $\leq 2^{k-2}$ total terms in the numerator and denominator must satisfy*

$$\int_{[0,1]} |f(x) - g(x)| \, \mathrm{d}x \geq \frac{1}{64}.$$

Note that this statement implies the desired difficulty of approximation, since a gap in the above integral (L_1) distance implies a gap in the earlier uniform distance (L_∞), and furthermore an r-degree rational function necessarily has $\leq 2r + 2$ total terms in its numerator and denominator.

As a final piece of the story, note that the conversion between rational functions and ReLU networks is more seamless if instead one converts to *rational networks*, meaning neural networks where each activation function is a rational function.

Lemma 1.3. *Let a ReLU network $f : [-1, +1]^d \to \mathbb{R}$ be given as in Theorem 1.1, meaning f has at most l layers and each node computes $z \mapsto \sigma_{\mathrm{r}}(a^\top z + b)$ where where the pair (a, b) (possibly distinct across nodes) satisfies $\|a\|_1 + |b| \leq 1$. Then there exists a rational function R of degree $\mathcal{O}(\ln(l/\epsilon)^2)$ so that replacing each σ_{r} in f with R yields a function $g : [-1, +1]^d \to \mathbb{R}$ with*

$$\sup_{x \in [-1,+1]^d} |f(x) - g(x)| \leq \epsilon.$$

Combining Theorem 1.2 and Lemma 1.3 yields an intriguing corollary.

Corollary 1.4. *For every $k \geq 3$, there exists a function $f : \mathbb{R} \to \mathbb{R}$ computed by a rational network with $\mathcal{O}(k)$ layers and $\mathcal{O}(k)$ total nodes, each node invoking a rational activation of degree $\mathcal{O}(k)$, such that every rational function $g : \mathbb{R} \to \mathbb{R}$ with less than 2^{k-2} total terms in the numerator and denominator satisfies*

$$\int_{[0,1]} |f(x) - g(x)| \, \mathrm{d}x \geq \frac{1}{128}.$$

The hard-to-approximate function f is a rational network which has a description of size $\mathcal{O}(k^2)$. Despite this, attempting to approximate it with a rational function of the usual form requires a description of size $\Omega(2^k)$. Said another way: even for rational functions, there is a benefit to a neural network representation!

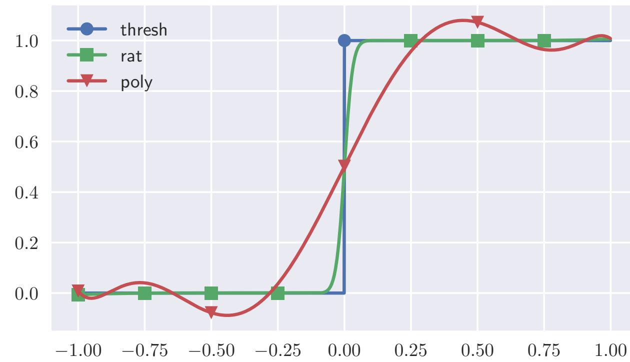

Figure 2. Polynomial and rational fit to the threshold function.

1.2. Auxiliary results

The first thing to stress is that Theorem 1.1 is impossible with polynomials: namely, while it is true that ReLU networks can efficiently approximate polynomials (Yarotsky, 2016; Safran & Shamir, 2016; Liang & Srikant, 2017), on the other hand polynomials require degree $\Omega(\text{poly}(1/\epsilon))$, rather than $\mathcal{O}(\text{poly}(\ln(1/\epsilon)))$, to approximate a single ReLU, or equivalently the absolute value function (Petrushev & Popov, 1987, Chapter 4, Page 73).

Another point of interest is the depth needed when converting a rational function to a ReLU network. Theorem 1.1 is impossible if the depth is $o(\ln(1/\epsilon))$: specifically, it is impossible to approximate the degree 1 rational function $x \mapsto 1/x$ with size $\mathcal{O}(\ln(1/\epsilon))$ but depth $o(\ln(1/\epsilon))$.

Proposition 1.5. *Set $f(x) := 1/x$, the reciprocal map. For any $\epsilon > 0$ and ReLU network $g : \mathbb{R} \to \mathbb{R}$ with l layers and $m < (27648\epsilon)^{-1/(2l)}/2$ nodes,*

$$\int_{[1/2,3/4]} |f(x) - g(x)| \, \mathrm{d}x > \epsilon.$$

Lastly, the implementation of division in a ReLU network requires a few steps, arguably the most interesting being a "continuous switch statement", which computes reciprocals differently based on the magnitude of the input. The ability to compute switch statements appears to be a fairly foundational operation available to neural networks and rational functions (Petrushev & Popov, 1987, Theorem 5.2), but is not available to polynomials (since otherwise they could approximate the ReLU).

1.3. Related work

The results of the present work follow a long line of work on the representation power of neural networks and related functions. The ability of ReLU networks to fit continuous functions was no doubt proved many times, but it appears the earliest reference is to Lebesgue (Newman, 1964, Page 1), though of course results of this type are usually given much more contemporary attribution (Cybenko, 1989). More recently, it has been shown that certain function classes only admit succinct representations with many layers (Telgarsky, 2015). This has been followed by proofs showing the possibility for a depth 3 function to require exponentially many nodes when rewritten with 2 layers (Eldan & Shamir, 2016). There are also a variety of other result giving the ability of ReLU networks to approximate various function classes (Cohen et al., 2016; Poggio et al., 2017).

Most recently, a variety of works pointed out neural networks can approximate polynomials, and thus smooth functions essentially by Taylor's theorem (Yarotsky, 2016; Safran & Shamir, 2016; Liang & Srikant, 2017). This somewhat motivates this present work, since polynomials can not in turn approximate neural networks with a dependence $\mathcal{O}(\text{poly}\log(1/\epsilon))$: they require degree $\Omega(1/\epsilon)$ even for a single ReLU.

Rational functions are extensively studied in the classical approximation theory literature (Lorentz et al., 1996; Petrushev & Popov, 1987). This literature draws close connections between rational functions and *splines* (piecewise polynomial functions), a connection which has been used in the machine learning literature to draw further connections to neural networks (Williamson & Bartlett, 1991). It is in this approximation theory literature that one can find the following astonishing fact: not only is it possible to approximate the absolute value function (and thus the ReLU) over $[-1,+1]$ to accuracy $\epsilon > 0$ with a rational function of degree $\mathcal{O}(\ln(1/\epsilon)^2)$ (Newman, 1964), but moreover the optimal rate is known (Petrushev & Popov, 1987; Zolotarev, 1877)! These results form the basis of those results here which show that rational functions can approximate ReLU networks. (Approximation theory results also provide other functions (and types of neural networks) which rational functions can approximate well, but the present work will stick to the ReLU for simplicity.)

An ICML reviewer revealed prior work which was embarrassingly overlooked by the author: it has been known, *since decades ago* (Beame et al., 1986), that neural networks using threshold nonlinearities (i.e., the map $x \mapsto \mathbb{1}[x \geq 0]$) can approximate division, and moreover the proof is similar to the proof of part 1 of Theorem 1.1! Moreover, other work on threshold networks invoked Newman polynomials to prove lower bound about linear threshold networks (Paturi & Saks, 1994). Together this suggests that not only the connections between rational functions and neural networks are tight (and somewhat known/unsurprising), but also that threshold networks and ReLU networks have perhaps more similarities than what is suggested by the differing VC dimension bounds, approximation results, and algorithmic results (Goel et al., 2017).

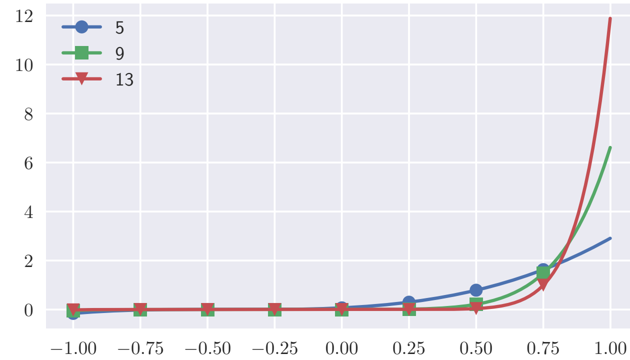

Figure 3. Newman polynomials of degree $5, 9, 13$.

1.4. Further notation

Here is a brief description of the sorts of neural networks used in this work. Neural networks represent computation as a directed graph, where nodes consume the outputs of their parents, apply a computation to them, and pass the resulting value onward. In the present work, nodes take their parents' outputs z and compute $\sigma_{\text{r}}(a^\top z + b)$, where a is a vector, b is a scalar, and $\sigma_{\text{r}}(x) := \max\{0, x\}$; another popular choice of nonlineary is the *sigmoid* $x \mapsto (1 + \exp(-x))^{-1}$. The graphs in the present work are acyclic and connected with a single node lacking children designated as the univariate output, but the literature contains many variations on all of these choices.

As stated previously, a rational function $f : \mathbb{R}^d \to \mathbb{R}$ is ratio of two polynomials. Following conventions in the approximation theory literature (Lorentz et al., 1996), the denominator polynomial will always be strictly positive. The degree of a rational function is the maximum of the degrees of its numerator and denominator.

2. Approximating ReLU networks with rational functions

This section will develop the proofs of part 2 of Theorem 1.1, Theorem 1.2, Lemma 1.3, and Corollary 1.4.

2.1. Newman polynomials

The starting point is a seminal result in the theory of rational functions (Zolotarev, 1877; Newman, 1964): there exists a rational function of degree $\mathcal{O}(\ln(1/\epsilon)^2)$ which can approximate the absolute value function along $[-1,+1]$ to accuracy $\epsilon > 0$. This in turn gives a way to approximate the ReLU, since

$$\sigma_{\text{r}}(x) = \max\{0, x\} = \frac{x + |x|}{2}. \tag{2.1}$$

The construction here uses the *Newman polynomials* (Newman, 1964): given an integer r, define

$$N_r(x) := \prod_{i=1}^{r-1} (x + \exp(-i/\sqrt{r})).$$

The Newman polynomials N_5, N_9, and N_{13} are depicted in Figure 3. Typical polynomials in approximation theory, for instance the Chebyshev polynomials, have very active oscillations; in comparison, the Newman polynomials look a little funny, lying close to 0 over $[-1, 0]$, and quickly increasing monotonically over $[0, 1]$. The seminal result of Newman (1964) is that

$$\sup_{|x|\leq 1} \left| |x| - x \left(\frac{N_r(x) - N_r(-x)}{N_r(x) + N_r(-x)} \right) \right| \leq 3\exp(-\sqrt{r})/2.$$

Thanks to this bound and eq. (2.1), it follows that the ReLU can be approximated to accuracy $\epsilon > 0$ by rational functions of degree $\mathcal{O}(\ln(1/\epsilon)^2)$.

(Some basics on Newman polynomials, as needed in the present work, can be found in Appendix A.1.)

2.2. Proof of Lemma 1.3

Now that a single ReLU can be easily converted to a rational function, the next task is to replace every ReLU in a ReLU network with a rational function, and compute the approximation error. This is precisely the statement of Lemma 1.3.

The proof of Lemma 1.3 is an induction on layers, with full details relegated to the appendix. The key computation, however, is as follows. Let $R(x)$ denote a rational approximation to σ_{r}. Fix a layer $i + 1$, and let $H(x)$ denote the multi-valued mapping computed by layer i, and let $H_R(x)$ denote the mapping obtained by replacing each σ_{r} in H with R. Fix any node in layer $i + 1$, and let $x \mapsto \sigma_{\mathrm{r}}(a^\top H(x) + b)$ denote its output as a function of the input. Then

$$\begin{aligned} &\left|\sigma_{\mathrm{r}}(a^\top H(x) + b) - R(a^\top H_R(x) + b)\right| \\ &\leq \underbrace{\left|\sigma_{\mathrm{r}}(a^\top H(x) + b) - \sigma_{\mathrm{r}}(a^\top H_R(x) + b)\right|}_{\heartsuit} \\ &\quad + \underbrace{\left|\sigma_{\mathrm{r}}(a^\top H_R(x) + b) - R(a^\top H_R(x) + b)\right|}_{\clubsuit}. \end{aligned}$$

For the first term $\heartsuit$, note since σ_{r} is 1-Lipschitz and by Hölder's inequality that

$$\heartsuit \leq \left| a^\top (H(x) - H_R(x)) \right| \leq \|a\|_1 \|H(x) - H_R(x)\|_\infty,$$

meaning this term has been reduced to the inductive hypothesis since $\|a\|_1 \leq 1$. For the second term $\clubsuit$, if $a^\top H_R(x) + b$ can be shown to lie in $[-1, +1]$ (which is another easy induction), then $\clubsuit$ is just the error between R and σ_{r} on the same input.

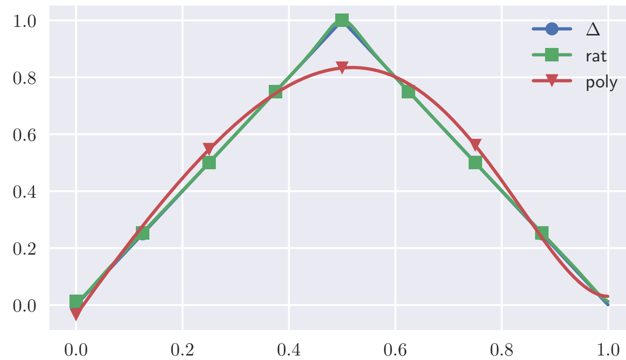

Figure 4. Polynomial and rational fit to Δ.

2.3. Proof of part 2 of Theorem 1.1

It is now easy to find a rational function that approximates a neural network, and to then bound its size. The first step, via Lemma 1.3, is to replace each σ_{r} with a rational function R of low degree (this last bit using Newman polynomials). The second step is to inductively collapse the network into a single rational function. The reason for the dependence on the number of nodes m is that, unlike polynomials, summing rational functions involves an increase in degree:

$$\frac{p_1(x)}{q_1(x)} + \frac{p_1(x)}{q_2(x)} = \frac{p_1(x)q_2(x) + p_2(x)q_1(x)}{q_1(x)q_2(x)}.$$

2.4. Proof of Theorem 1.2

The final interesting bit is to show that the dependence on m^l in part 2 of Theorem 1.1 (where m is the number of nodes and l is the number of layers) is tight.

Recall the "triangle function"

$$\Delta(x) := \begin{cases} 2x & x \in [0, 1/2], \\ 2(1-x) & x \in (1/2, 1], \\ 0 & \text{otherwise.} \end{cases}$$

The k-fold composition Δ^k is a piecewise affine function with 2^{k-1} regularly spaced peaks (Telgarsky, 2015). This function was demonstrated to be inapproximable by shallow networks of subexponential size, and now it can be shown to be a hard case for rational approximation as well.

Consider the horizontal line through $y = 1/2$. The function Δ^k will cross this line 2^k times. Now consider a rational function $f(x) = p(x)/q(x)$. The set of points where $f(x) = 1/2$ corresponds to points where $2p(x) - q(x) = 0$.

A poor estimate for the number of zeros is simply the degree of $2p-q$, however, since f is univariate, a stronger tool becomes available: by Descartes' rule of signs, the number of zeros in $f - 1/2$ is upper bounded by the number of terms in $2p - q$.

3. Approximating rational functions with ReLU networks

This section will develop the proof of part 1 of Theorem 1.1, as well as the tightness result in Proposition 1.5

3.1. Proving part 1 of Theorem 1.1

To establish part 1 of Theorem 1.1, the first step is to approximate polynomials with ReLU networks, and the second is to then approximate the division operation.

The representation of polynomials will be based upon constructions due to Yarotsky (2016). The starting point is the following approximation of the squaring function.

Lemma 3.1 ((Yarotsky, 2016)). *Let any $\epsilon > 0$ be given. There exists $f : x \to [0,1]$, represented as a ReLU network with $\mathcal{O}(\ln(1/\epsilon))$ nodes and layers, such that $\sup_{x\in[0,1]} |f(x) - x^2| \le \epsilon$ and $f(0) = 0$.*

Yarotsky's proof is beautiful and deserves mention. The approximation of x^2 is the function f_k, defined as

$$f_k(x) := x - \sum_{i=1}^{k} \frac{\Delta^i(x)}{4^i},$$

where Δ is the triangle map from Section 2. For every k, f_k is a convex, piecewise-affine interpolation between points along the graph of x^2; going from k to $k+1$ does not adjust any of these interpolation points, but adds a new set of $\mathcal{O}(2^k)$ interpolation points.

Once squaring is in place, multiplication comes via the polarization identity $xy = ((x+y)^2 - x^2 - y^2)/2$.

Lemma 3.2 ((Yarotsky, 2016)). *Let any $\epsilon > 0$ and $B \ge 1$ be given. There exists $g(x,y) : [0,B]^2 \to [0,B^2]$, represented by a ReLU network with $\mathcal{O}(\ln(B/\epsilon)$ nodes and layers, with*

$$\sup_{x,y\in[0,1]} |g(x,y) - xy| \le \epsilon$$

and $g(x,y) = 0$ if $x = 0$ or $y = 0$.

Next, it follows that ReLU networks can efficiently approximate exponentiation thanks to repeated squaring.

Lemma 3.3. *Let $\epsilon \in (0,1]$ and positive integer y be given. There exists $h : [0,1] \to [0,1]$, represented by a ReLU network with $\mathcal{O}(\ln(y/\epsilon)^2)$ nodes and layers, with*

$$\sup_{x,y\in[0,1]} \left|h(x) - x^y\right| \le \epsilon$$

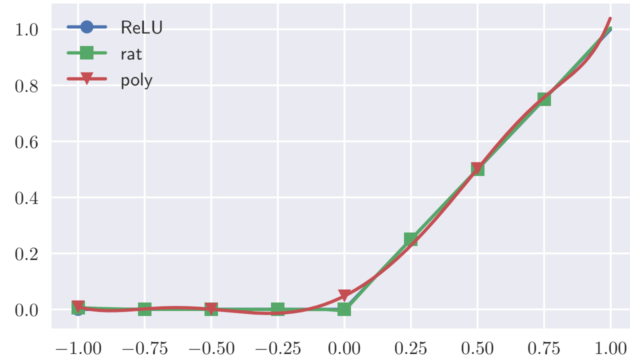

Figure 5. Polynomial and rational fit to σ_{r}.

With multiplication and exponentiation, a representation result for polynomials follows.

Lemma 3.4. *Let $\epsilon \in (0,1]$ be given. Let $p : [0,1]^d \to [-1,+1]$ denote a polynomial with $\le s$ monomials, each with degree $\le r$ and scalar coefficient within $[-1,+1]$. Then there exists a function $q : [0,1]^d \to [-1,+1]$ computed by a network of size $\mathcal{O}\left(\min\{sr\ln(sr/\epsilon), sd\ln(dsr/\epsilon)^2\}\right)$, which satisfies $\sup_{x\in[0,1]^d} |p(x) - q(x)| \le \epsilon$.*

The remainder of the proof now focuses on the division operation. Since multiplication has been handled, it suffices to compute a single reciprocal.

Lemma 3.5. *Let $\epsilon \in (0,1]$ and nonnegative integer k be given. There exists a ReLU network $q : [2^{-k},1] \to [1,2^k]$, of size $\mathcal{O}(k^2\ln(1/\epsilon)^2)$ and depth $\mathcal{O}(k^4\ln(1/\epsilon)^3)$ such that*

$$\sup_{x\in[2^{-k},1]} \left|q(x) - \frac{1}{x}\right| \le \epsilon.$$

This proof relies on two tricks. The first is to observe, for $x \in (0,1]$, that

$$\frac{1}{x} = \frac{1}{1-(1-x)} = \sum_{i\ge 0}(1-x)^i.$$

Thanks to the earlier development of exponentiation, truncating this summation gives an expression easily approximate by a neural network as follows.

Lemma 3.6. *Let $0 < a \le b$ and $\epsilon > 0$ be given. Then there exists a ReLU network $q : \mathbb{R} \to \mathbb{R}$ with $\mathcal{O}(\ln(1/(a\epsilon))^2)$ layers and $\mathcal{O}((b/a)\ln(1/(a\epsilon))^3)$ nodes satisfying*

$$\sup_{x\in[a,b]} \left|q(x) - \frac{1}{x}\right| \le 2\epsilon.$$

Unfortunately, Lemma 3.6 differs from the desired statement Lemma 3.6: inverting inputs lying within $[2^{-k},1]$ requires $\mathcal{O}(2^k\ln(1/\epsilon)^2)$ nodes rather than $\mathcal{O}(k^4\ln(1/\epsilon)^3)$!

To obtain a good estimate with only $\mathcal{O}(\ln(1/\epsilon))$ terms of the summation, it is necessary for the input to be x bounded below by a positive constant (not depending on k). This leads to the second trick (which was also used by Beame et al. (1986)!).

Consider, for positive constant $c > 0$, the expression

$$\frac{1}{x} = \frac{c}{1-(1-cx)} = c\sum_{i\geq 0}(1-cx)^i.$$

If x is small, choosing a larger c will cause this summation to converge more quickly. Thus, to compute $1/x$ accurately over a wide range of inputs, the solution here is to multiplex approximations of the truncated sum for *many* choices of c. In order to only rely on the value of one of them, it is possible to encode a large "switch" style statement in a neural network. Notably, rational functions can also represantat switch statements (Petrushev & Popov, 1987, Theorem 5.2), however polynomials can not (otherwise they could approximate the ReLU more efficiently, seeing as it is a switch statement of 0 (a degree 0 polynomial) and x (a degree 1 polynomial).

Lemma 3.7. *Let $\epsilon > 0$, $B \geq 1$, reals $a_0 \leq a_1 \leq \cdots \leq a_n \leq a_{n+1}$ and a function $f : [a_0, a_{n+1}] \to \mathbb{R}$ be given. Moreover, suppose for $i \in \{1, \ldots, n\}$, there exists a ReLU network $g_i : \mathbb{R} \to \mathbb{R}$ of size $\leq m_i$ and depth $\leq k_i$ with $g_i \in [0, B]$ along $[a_{i-1}, a_{i+1}]$ and*

$$\sup_{x\in[a_{i-1},a_{i+1}]} |g_i(x) - f| \leq \epsilon.$$

Then there exists a function $g : \mathbb{R} \to \mathbb{R}$ computed by a ReLU network of size $\mathcal{O}\left(n \ln(B/\epsilon) + \sum_i m_i\right)$ and depth $\mathcal{O}\left(\ln(B/\epsilon) + \max_i k_i\right)$ satisfying

$$\sup_{x\in[a_1,a_n]} |g(x) - f(x)| \leq 3\epsilon.$$

3.2. Proof of Proposition 1.5

It remains to show that shallow networks have a hard time approximating the reciprocal map $x \mapsto 1/x$.

This proof uses the same scheme as various proofs in (Telgarsky, 2016), which was also followed in more recent works (Yarotsky, 2016; Safran & Shamir, 2016): the idea is to first upper bound the number of affine pieces in ReLU networks of a certain size, and then to point out that each linear segment must make substantial error on a curved function, namely $1/x$.

The proof is fairly brute force, and thus relegated to the appendices.

4. Summary of figures

Throughout this work, a number of figures were presented to show not only the astonishing approximation properties of rational functions, but also the higher fidelity approximation achieved by both ReLU networks and rational functions as compared with polynomials. Of course, this is only a qualitative demonstration, but still lends some intuition.

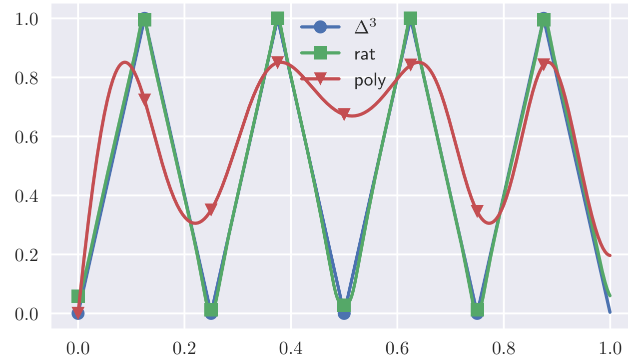

Figure 6. Polynomial and rational fit to Δ^3.

In all these demonstrations, rational functions and polynomials have degree 9 unless otherwise marked. ReLU networks have two hidden layers each with 3 nodes. This is not exactly apples to apples (e.g., the rational function has twice as many parameters as the polynomial), but still reasonable as most of the approximation literature fixes polynomial and rational degrees in comparisons.

Figure 1 shows the ability of all three classes to approximate a truncated reciprocal. Both rational functions and ReLU networks have the ability to form "switch statements" that let them approximate different functions on different intervals with low complexity (Petrushev & Popov, 1987, Theorem 5.2). Polynomials lack this ability; they can not even approximate the ReLU well, despite it being low degree polynomials on two separate intervals.

Figure 2 shows that rational functions can fit the threshold function errily well; the particular rational function used here is based on using Newman polynomials to approximate $(1 + |x|/x)/2$ (Newman, 1964).

Figure 3 shows Newman polynomials N_5, N_9, N_{13}. As discussed in the text, they are unlike orthogonal polynomials, and are used in all rational function approximations except Figure 1, which used a least squares fit.

Figure 4 shows that rational functions (via the Newman polynomials) fit Δ very well, whereas polynomials have trouble. These errors degrade sharply after recursing, namely when approximating Δ^3 as in Figure 6.

Figure 5 shows how polynomials and rational functions fit the ReLU, where the ReLU representation, based on Newman polynomials, is the one used in the proofs here. Despite the apparent slow convergence of polynomials in this regime, the polynomial fit is still quite respectable.

5. Open problems

There are many next steps for this and related results.

1. Can rational functions, or some other approximating class, be used to more tightly bound the generalization properties of neural networks? Notably, the VC dimension of sigmoid networks uses a conversion to polynomials (Anthony & Bartlett, 1999).
2. Can rational functions, or some other approximating class, be used to design algorithms for training neural networks? It does not seem easy to design reasonable algorithms for minimization over rational functions; if this is fundamental and moreover in contrast with neural networks, it suggests an algorithmic benefit of neural networks.
3. Can rational functions, or some other approximating class, give a sufficiently refined complexity estimate of neural networks which can then be turned into a regularization scheme for neural networks?

Acknowledgements

The author thanks Adam Klivans and Suvrit Sra for stimulating conversations. Adam Klivans and the author both thank Almare Gelato Italiano, in downtown Berkeley, for necessitating further stimulating conversations, but now on the topic of health and exercise. Lastly, the author thanks the University of Illinois, Urbana-Champaign, and the Simons Institute in Berkeley, for financial support during this work.

References

Anthony, Martin and Bartlett, Peter L. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

Beame, Paul, Cook, Stephen A., and Hoover, H. James. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, 1986.

Cohen, Nadav, Sharir, Or, and Shashua, Amnon. On the expressive power of deep learning: A tensor analysis. 2016. COLT.

Cybenko, George. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Eldan, Ronen and Shamir, Ohad. The power of depth for feedforward neural networks. In *COLT*, 2016.

Goel, Surbhi, Kanade, Varun, Klivans, Adam, and Thaler, Justin. Reliably learning the relu in polynomial time. In *COLT*, 2017.

Liang, Shiyu and Srikant, R. Why deep neural networks for function approximation? In *ICLR*, 2017.

Lorentz, G. G., Golitschek, Manfred von, and Makovoz, Yuly. *Constructive approximation : advanced problems*. Springer, 1996.

Newman, D. J. Rational approximation to $|x|$. *Michigan Math. J.*, 11(1):11–14, 03 1964.

Paturi, Ramamohan and Saks, Michael E. Approximating threshold circuits by rational functions. *Inf. Comput.*, 112(2):257–272, 1994.

Petrushev, P. P. Penco Petrov and Popov, Vasil A. *Rational approximation of real functions*. Encyclopedia of mathematics and its applications. Cambridge University Press, 1987.

Poggio, Tomaso, Mhaskar, Hrushikesh, Rosasco, Lorenzo, Miranda, Brando, and Liao, Qianli. Why and when can deep – but not shallow – networks avoid the curse of dimensionality: a review. 2017. `arXiv:1611.00740 [cs.LG]`.

Safran, Itay and Shamir, Ohad. Depth separation in relu networks for approximating smooth non-linear functions. 2016. `arXiv:1610.09887 [cs.LG]`.

Telgarsky, Matus. Representation benefits of deep feedforward networks. 2015. `arXiv:1509.08101v2 [cs.LG]`.

Telgarsky, Matus. Benefits of depth in neural networks. In *COLT*, 2016.

Williamson, Robert C. and Bartlett, Peter L. Splines, rational functions and neural networks. In *NIPS*, 1991.

Yarotsky, Dmitry. Error bounds for approximations with deep relu networks. 2016. `arXiv:1610.01145 [cs.LG]`.

Zolotarev, E.I. Application of elliptic functions to the problem of the functions of the least and most deviation from zero. *Transactions Russian Acad. Scai.*, pp. 221, 1877.

A. Deferred material from Section 2

This section collects technical material omitted from Section 2. The first step is to fill in some missing details regarding Newman polynomials.

A.1. Newman polynomials

Define the *Newman polynomial* (Newman, 1964)

$$N_r(x) := \prod_{i=1}^{r-1}(x+\alpha_r^i) \qquad \text{where } \alpha_r := \exp(-1/\sqrt{r}). \tag{A.1}$$

Define $A_r(x)$, the Newman approximation to $|x|$, as

$$A_r(x) := x\left(\frac{N_r(x)-N_r(-x)}{N_r(x)+N_r(-x)}\right).$$

Lemma A.2 (Newman (1964))**.** *Suppose* $r \geq 5$.

- $N_r(x)+N_r(-x) > 0$*; in particular,* A_r *is well-defined over* $\mathbb{R}$.
- *Given any* $b \geq 1$,
$$\sup_{x\in[-b,+b]} \left|bA_r(x/b) - |x|\right| \leq 3b\exp(-\sqrt{r}).$$

Proof.

- If $x = 0$, then $N_r(-x) = N_r(x) = \prod_{i=1}^{r-1}\alpha_r^i > 0$. Otherwise $x > 0$, and note for any $i \in \{1,\ldots,r-1\}$ that
  - $x \in (0, \alpha_r^i]$ means $|x-\alpha_r^i| = \alpha_r^i - x < \alpha_r^i + x$,
  - $x > \alpha_r^i$ means $|x-\alpha_r^i| = x - \alpha_r^i < x + \alpha_r^i$.

  Together, $|x-\alpha_r^i| < x+\alpha_r^i$, and
  $$N_r(x) = \prod_{i=1}^{r-1}(x+\alpha_r^i) > \prod_{i=1}^{r-1}|x-\alpha_r^i| = \left|\prod_{i=1}^{r-1}(x-\alpha_r^i)\right| = |N_r(-x)|.$$
  Since $N_r(x) > 0$ when $x > 0$, thus $N_r(x)+N_r(-x) > N_r(x) - |N_r(x)| = 0$.

  Lastly, the case $x < 0$ follows from the case $x > 0$ since $x \mapsto N_r(x)+N_r(-x)$ is even.
- For any $x \in [-b,+b]$,
  $$\left||x| - bA_r(x/b)\right| = \left|b\left(|x/b| - A_r(x/b)\right)\right| = b\left|x/b - A_r(x/b)\right| \leq 3b\exp(-\sqrt{r}),$$
  where the last step was proved by Newman (Lorentz et al., 1996, Theorem 7.3.1).

□

Finally, define

$$\begin{aligned}
\tilde{R}_{r,b}(x) := R_r(x;b) &:= \frac{x+bA_r(x/b)}{2},\\
\epsilon_{r,b} &:= 3\exp(-\sqrt{r})/2,\\
R_{r,b}(x) &:= (1-2\epsilon_{r,b})\tilde{R}_{r,b}(x) + b\epsilon_{r,b}.
\end{aligned}$$

Lemma A.3. *If $r \geq 5$ and $b \geq 1$ and $\epsilon_{r,b} \leq 1/2$, then $R_{r,b}$ is a degree-r rational function over $\mathbb{R}$, and*

$$\sup_{x\in[-b,+b]}\left|\sigma_{\mathrm{r}}(x) - \tilde{R}_{r,b}(x)\right| \leq b\epsilon_{r,b},$$
$$\sup_{x\in[-b,+b]}\left|\sigma_{\mathrm{r}}(x) - R_{r,b}(x)\right| \leq 3b\epsilon_{r,b}.$$

If $\epsilon_{r,b} \leq 1$, then $R_{r,b} \in [0, b]$ along $[-b, +b]$.

Proof. Let r, b be given, and for simplicity omit the various subscripts. The denominator of $\tilde{R}$ is positive over $\mathbb{R}$ by Lemma A.2. Now fix $x \in [-b, +b]$. Using the second part of Lemma A.2,

$$\left|\sigma_{\mathrm{r}}(x) - \tilde{R}(x)\right| = \left|\frac{x + |x|}{2} - \frac{x + bA(x/b)}{2}\right| = \frac{1}{2}\left||x| - bA(x/b)\right| \leq 3b\exp(-\sqrt{r})/2 = b\epsilon.$$

Next, note that $\tilde{R} \in [-b\epsilon, b(1+\epsilon)]$:

$$\tilde{R}(x) \leq \sigma_{\mathrm{r}}(x) + b\epsilon \leq b(1+\epsilon), \qquad \tilde{R}(x) \geq \sigma_{\mathrm{r}}(x) - b\epsilon \geq -b\epsilon.$$

Thus

$$\begin{aligned}\left|\sigma_{\mathrm{r}}(x) - R(x)\right| &\leq \left|\sigma_{\mathrm{r}}(x) - \tilde{R}(x)\right| + \left|\tilde{R}(x) - R(x)\right| \\ &\leq b\epsilon + 0 + 2\epsilon\left|\tilde{R}(x) - b/2\right| \\ &\leq 3b\epsilon.\end{aligned}$$

Moreover

$$\begin{aligned}&R(x) = (1-2\epsilon)\tilde{R}(x) + b\epsilon \geq (1-2\epsilon)(-b\epsilon) + b\epsilon \geq 0, \\ &R(x) \leq (1-2\epsilon)b(1+\epsilon) + b\epsilon \leq b.\end{aligned}$$

□

A.2. Remaining deferred proofs

The details of converting a ReLU network into a rational network are as follows.

Lemma A.4. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be represented by a ReLU network with $\leq l$ layers, and with each node computing a map $z \mapsto \sigma_{\mathrm{r}}(a^\top z + b)$ where $\|a\|_1 + |b| \leq 1$. Then for every $\epsilon > 0$ there exists a function $g : \mathbb{R}^d \to \mathbb{R}$ with $|g(x) - f(x)| \leq \epsilon$ for $\|x\|_\infty \leq 1$ where g is obtained from f by replacing each ReLU with an r-rational function with $r = \mathcal{O}(\ln(1/\epsilon)^2)$.*

Proof of Lemma 1.3. This construction will use the Newman-based approximation $R := R_{r,b}$ to σ_{r} with degree $\mathcal{O}(\ln(l/\epsilon)^2)$. By Lemma A.3, this degree suffices to guarantee $R(x) \in [0, 1]$ and $|R(x) - \sigma_{\mathrm{r}}(x)| \leq \epsilon/l$ for $|x| \leq 1$.

First note, by induction on layers, that the output of every node has absolute value at most 1. The base case is the inputs themselves, and thus the statement holds by the assumption $\|x\|_\infty \leq 1$. In the inductive step, consider any node $z \mapsto R(a^\top z + b)$, where z is the multivariate input to this node. By the inductive hypothesis, $\|z\|_\infty \leq 1$, thus

$$|a^\top z + b| \leq \|a\|_1\|z\|_\infty + |b| \leq 1.$$

As such, $R(a^\top z + b) \in [0, 1]$.

It remains to prove the error bound. For any node, if $h : \mathbb{R}^d \to \mathbb{R}$ denote the function (of the input x) compute by this node, then let h_R denote the function obtained by replacing all ReLUs with R. It will be shown that every node in layer i has $|h_R(x) - h(x)| \leq i\epsilon/l$ when $\|x\|_\infty \leq 1$. The base case is the inputs themselves, and thus there is no approximation error, meaning the bound holds with error $0 \leq 1 \cdot \epsilon/l$. Now consider any node in layer $i+1$ with $i \geq 0$, and suppose the claim holds for nodes in layers i and lower. For convenience, let H denote the multivalued map computed by the previous layer, and H_R denote the multivalued map obtained by replacing all activations in earlier layers with R. Since σ_{r} is 1-Lipschitz, and since the earlier boundedness property grants

$$\left|a^\top H_R(x) + b\right| \leq \|a\|_1 \|H_R(x)\|_\infty + |b| \leq 1,$$

then

$$\begin{aligned}
|h(x) - h_R(x)| &= \left|\sigma_{\mathrm{r}}(a^\top H(x) + b) - R(a^\top H_R(x) + b)\right| \\
&\leq \left|\sigma_{\mathrm{r}}(a^\top H(x) + b) - \sigma_{\mathrm{r}}(a^\top H_R(x) + b)\right| + \left|\sigma_{\mathrm{r}}(a^\top H_R(x) + b) - R(a^\top H_R(x) + b)\right| \\
&\leq \left|a^\top H(x) - a^\top H_R(x)\right| + \epsilon/l \\
&\leq \|a\|_1 \|H - H_R\|_\infty + \epsilon/l \\
&\leq (i+1)\epsilon/l.
\end{aligned}$$

□

Next, collapsing a rational network down into a single rational function is proved as follows.

Lemma A.5. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a rational network with $\leq m$ nodes in each of $\leq l$ layers, and the activation function has degree r. Then the rational function obtained by collapsing f has degree at most $(rm)^l$.*

Proof. Throughout this proof, let R denote the rational activation function at each node, and write $R(x) = p(x)/q(x)$ where p and q are polynomials of degree at most r. The proof establishes, by induction on layers, that the nodes of layer i compute rational functions of degree at most $(rm)^i$. The base case is layer 1, where each node computes a rational function of degree $r \leq rm$. For the case of layer $i > 1$, fix any node, and denote its computation by $h(x) = R(\sum_{j=1}^n a_j g_j(x) + b)$, where $n \leq m$ and $g_j = p_j/q_j$ is a rational function of degree at most $(rm)^{i-1}$. Note

$$\begin{aligned}
\deg\left(\sum_j \frac{a_j p_j(x)}{q_j(x)} + b\right) &= \deg\left(\frac{b \prod_j q_j(x) + \sum_j a_j p_j(x) \prod_{k \neq j} q_k(x)}{\prod_j q_j(x)}\right) \\
&\leq m(mr)^{i-1}.
\end{aligned}$$

the map $f := \sum_j a_j g_j + b$ is rational of degree $m(mr)^{i-1}$. Let p_f and q_f denote its numerator and denominator. Since R is univariate, its numerator p and denominator q have the form $p(x) := \sum_{j \leq r} c_j x^j$ and $q(x) \sum_{j \leq r} d_j x^j$. Thus, using the fact that $q > 0$,

$$\begin{aligned}
\deg(h(x)) = \deg(R(f(x))) &= \deg\left(\frac{\sum_{j \leq r} c_j (p_f(x)/q_f(x))^j}{\sum_{j \leq r} d_j (p_f(x)/q_f(x))^j}\left(\frac{q_f(x)^r}{q_f(x)^r}\right)\right) \\
&= \deg\left(\frac{\sum_{j \leq r} c_j p_f(x)^j q_f(x))^{r-j}}{\sum_{j \leq r} d_j p_f(x)^j q_f(x))^{r-j}}\right) \leq rm(rm)^{i-1} = (rm)^i.
\end{aligned}$$

□

The proof of part 2 of Theorem 1.1 now follows by combining Lemmas 1.3, A.3 and A.5.

The last piece is a slighly more detailed account of Theorem 1.2.

Proof of Theorem 1.2. Let $\Delta : \mathbb{R} \to \mathbb{R}$ denote the triangle function from (Telgarsky, 2015):

$$\Delta(x) := \begin{cases} 2x & x \in [0, 1/2], \\ 2(1-x) & x \in (1/2, 1], \\ 0 & \text{otherwise.} \end{cases}$$

Define the target function $f = \Delta^k$, which as in (Telgarsky, 2015) has 2^k regular-spaced crossings of 0 along $[0,1]$, and can be written as a network with $2k$ layers, each with ≤ 2 nodes.

Next consider the rational function g. As in the text, it is necessary to count the zeros of $g - 1/2$ (the case $g = 1/2$ is trivial). Writing $g = p/q$, equivalently this means the zeros of $2p - q$. Since p and q together have $\leq 2^{k-2}$ terms, by Descartes' rule of signs, g crosses $1/2$ at most 2^{k-2} times along $(0,1]$. Therefore, following a similar calculation to the proof in (Telgarsky, 2016, Proof of Theorem 1.1),

$$\int_{(0,1]} |f(x) - g(x)| \, \mathrm{d}x \geq \frac{1}{32}\left(1 - \frac{2(2^{k-2})}{2^k}\right) = \frac{1}{64}.$$

□

B. Deferred material from Section 3

B.1. Towards the proof of part 1 of Theorem 1.1

To start, the lemmas due to Yarotsky are slightly adjusted to clip the range to $[0,1]$.

Proof of Lemma 3.1. Inspecting Yarotsky's proof, the construction provides $g(x)$ with $g(0) = 0$ and $\sup_{x\in[0,1]} |g(x) - x^2| \leq \epsilon$. To provide the desired f, it suffices to define $f(x) = \sigma_{\mathrm{r}}(g(x)) - \sigma_{\mathrm{r}}(g(x) - 1)$. □

Proof of Lemma 3.2. First suppose $B = 1$, let f be as in Lemma 3.1 at resolution $\epsilon/8$, and define h via the polarization identity (as in Yarotsky's proof):

$$h(x,y) = 2(f(x/2 + y/2) - f(x/2) - f(y/2))$$

(where $x/2$ appears since f has domain $[0,1]^2$). Since $f(0) = 0$,

$$h(x,0) = 2(f(x/2) - f(x/2) - 0) = 0, \qquad h(0,y) = 2(f(y/2) - 0 - f(y/2)) = 0.$$

Moreover, for any $x, y \in [0,1]$

$$h(x,y) - xy \leq 2\left((x/2 + y/2)^2 + \epsilon/8 - x^2/4 + \epsilon/8 - y^2/4 + \epsilon/8\right) - xy \leq xy + \epsilon,$$
$$h(x,y) - xy \geq 2\left((x/2 + y/2)^2 - \epsilon/8 - x^2/4 - \epsilon/8 - y^2/4 - \epsilon/8\right) - xy \leq xy - \epsilon.$$

Finally, set $\tilde{g}(x,y) := \sigma_{\mathrm{r}}(h(x,y)) - \sigma_{\mathrm{r}}(h(x,y) - 1)$, which preserves the other properties.

Now consider the case $B \geq 1$, and set $g(x,y) = B^2 g(x/B, y/B)$. Then $(x,y) \in [0,B]^2$ implies

$$\left|g(x,y) - xy\right| = B^2\left|\tilde{g}(x/B, y/B) - (x/B)(y/B)\right| \leq \epsilon B^2,$$

and $g \in [0, B^2]$ over $[0,B]^2$ since $\tilde{g} \in [0,1]$ over $[0,1]^2$. □

The full details for the proof of fast exponentiation are as follows.

Proof of Lemma 3.3. This proof constructs a network implementing the *russian peasant algorithm for exponentiation*:

1. Set $v := 1$.
2. For $b \in$ bits-ltr(y) (the bits of y from left to right):
   (a) Set $v := v^2$.
   (b) If $b = 1$, set $v := vx$.

For example,

$$x^{10101_2} = ((((1^2 \cdot x)^2)^2 \cdot x)^2)^2 \cdot x = x^{2^4} \cdot x^{2^2} \cdot x.$$

The two lines in the inner loop will use the squaring function f from Lemma 3.1 and the multiplication function g from Lemma 3.2, each with accuracy $c\epsilon$ where $c := 1/y^2$. At the end, the network returns $\sigma_{\mathrm{r}}(v) - \sigma_{\mathrm{r}}(v-1)$ to ensure the output lies in $[0,1]$; this procedure can not increase the error. Since the loop is invoke $\mathcal{O}(\ln(y))$ times and each inner loop requires a network of size $\mathcal{O}(\ln(1/(c\epsilon))) = \mathcal{O}(\ln(y/\epsilon))$, the full network has size $\mathcal{O}(\ln(y/\epsilon)^2)$.

It remains to show that the network computes a function h which satisfies

$$h(x) = x^y.$$

Let z_j denote the integer corresponding to the first j bits of y when read left-to-right; it will be shown by induction (on the bits of y from left to right) that, at the end of the jth invocation of the loop,

$$|v - x^{z_j}| \leq z_j^2 c\epsilon.$$

This suffices to establish the claim since then $|v - x^y| \leq y^2 c\epsilon = \epsilon$.

For the base case, consider $j = 0$; then $v = 1 = x^{z_0} = x^0$ as desired. For the inductive step, let w denote v at the end of the previous iteration, whereby the inductive hypothesis grants

$$|w - x^{z_{j-1}}| \leq z_{j-1}^2 c\epsilon.$$

The error after the approximate squaring step can be upper bounded as

$$\begin{aligned}
f(w) - x^{2z_{j-1}} &\leq \Big(f(w) - w^2\Big) + \Big(w^2 - x^{2z_{j-1}}\Big) \\
&\leq c\epsilon + \Big((x^{z_{j-1}} + z_{j-1}^2 c\epsilon)^2 - x^{2z_{j-1}}\Big) \\
&\leq c\epsilon + 2z_{j-1}^2 c\epsilon + z_{j-1}^4 c^2\epsilon^2 \\
&\leq c\epsilon + 2z_{j-1}^2 c\epsilon + z_{j-1}^2 c\epsilon \\
&\leq (2z_{j-1})^2 c\epsilon.
\end{aligned}$$

The reverse inequality is proved analogously, thus

$$\Big|f(w) - x^{2z_{j-1}}\Big| \leq (2z_{j-1})^2 c\epsilon.$$

If the bit b in this iteration is 0, then $2z_{j-1} = z_j$ and the proof for this loop iteration is complete. Otherwise $b = 1$, and

$$\begin{aligned}
v - x^{z_j} &= g(f(w), x) - x^{2z_{j-1}+b} \\
&\leq xf(w) + c\epsilon - x^{2z_{j-1}+b} \\
&\leq \Big((2z_{j-1})^2 + 1\Big) c\epsilon \\
&\leq (z_j)^2 c\epsilon.
\end{aligned}$$

The proof of the reverse inequality is analogous, which establishes the desired error bound on v for this loop iteration. $\square$

Using the preceding exponentiation lemma, the proof of polynomial approximation is as follows.

Proof of Lemma 3.4. It will be shown momentarily that a single monomial term can be approximating to accuracy ϵ/s with a network of size $\mathcal{O}\left(\min\{r\ln(sr/\epsilon), d\ln(dsr/\epsilon)^2\}\right)$. This implies the result by summing $\leq s$ monomials comprising a polynomial, along with their errors.

For a single monomial, here are two constructions.

- One approach is to product together $\leq r$ individual variables (and lastly multiple by a fixed scalar coefficient), with no concern of the multiplicities of individual variables. To this end, let $(y_1, \ldots, y_k)$ with $s \leq r$ denote coordinates of the input variable so that $\prod_{i=1}^{k} y_i$ is the desired multinomial. Let g denote multiplication with error $\epsilon_0 := \epsilon/(rs)$ as provided by Lemma 3.2. The network will compute $\alpha g_i(y)$, where $\alpha \in [-1, +1]$ is the scalar coefficient on the monomial, and g_i is recursively defined as

$$g_1(y) = y_1, \qquad g_{i+1}(y) := f(y_{i+1}, g_i(y))$$

It is established by induction that

$$\left| g_i(y) - \prod_{j=1}^{i} y_j \right| \leq j\epsilon_0$$

The base case is immediate since $g_1(y) = y_1 = \prod_{j=1}^{1} y_j$. For the inductive step,

$$g_{i+1}(y) - \prod_{j=1}^{i+1} y_j = f(y_{i+1}, g_i(y)) - \prod_{j=1}^{i+1} y_j \leq y_{i+1} g_i(y) + \epsilon_0 - \prod_{j=1}^{i+1} y_j \leq y_{i+1}(i\epsilon_0 + \prod_{j=1}^{i} y_j) + \epsilon_0 - \prod_{j=1}^{i+1} y_j \leq (i+1)\epsilon_0,$$

and the reverse inequality is proved analogously.

- Alternatively, the network uses the fast exponentiation routine from Lemma 3.3, and then multiplies together the terms for individual coordinates. In particular, the exponentiation for each coordinate with accuracy $\epsilon_1 := \epsilon/(ds)$ requires a network of size $\mathcal{O}(\ln(r/\epsilon_1)^2)$. By an analysis similar to the preceding construction, multiplying $\leq d$ such networks will result in a network approximating the monomial with error ϵ/s and size $\mathcal{O}(d \ln(r/\epsilon_1)^2)$.

$\square$

Next, the proof that ReLU networks can efficiently compute reciprocals, namely Lemma 3.5. As stated in the text, it is first necessary to establish Lemma 3.6, which gives computes reciprocals at a choice of magnitude, and then Lemma 3.7, which combines these circuits across scales.

Proof of Lemma 3.7. For each $i \in \{1, \ldots, n\}$, define the function

$$p_i(z) := \begin{cases} \frac{z - a_{i-1}}{a_i - a_{i-1}} & z \in [a_{i-1}, a_i], \\ \frac{a_{i+1} - z}{a_{i+1} - a_i} & z \in (a_i, a_{i+1}], \\ 0 & \text{otherwise.} \end{cases}$$

The functions $(p_i)_{i=1}^{n}$ have the following properties.

- Each p_i can be represented by a ReLU network with three nodes in 2 layers.
- For any $x \in [a_1, a_n]$, there exists $j \in \{1, \ldots, n\}$ so that $i \in \{j, j+1\}$ implies $p_i(x) \geq 0$ and $i \notin \{j, j+1\}$ implies $p_i(x) = 0$. Indeed, it suffices to let j be the smallest element of $\{1, \ldots, n-1\}$. satisfying $x \in [a_j, a_{j+1}]$.
- For any $x \in [a_1, a_n]$, $\sum_{i=1}^{n} p_i(x) = 1$.

The family $(p_i)_{i=1}^{n}$ thus forms a *partition of unity* over $[a_1, a_n]$, moreover with the property that at most two elements, necessarily consecutive, are nonzero at any point in the interval.

Let $h : [0, B]^2 \to [0, B]$ be a uniform ϵ-approximation via ReLU networks to the multiplication map $(x, y) \mapsto xy$; by Lemma 3.2, h has $\mathcal{O}(\ln(B/\epsilon))$ nodes and layers, and moreover the multiplication is exact when either input is 0. Finally, define $g : \mathbb{R} \to \mathbb{R}$ as

$$g(x) := \sum_{i=1}^{n} h(p_i(x), g_i(x)).$$

By construction, g is a ReLU network with $\mathcal{O}(\ln(B/\epsilon) + \max_i k_i)$ layers and $\mathcal{O}(n \ln(B/\epsilon) + \sum_i m_i)$ nodes.

It remains to check the approximation properties of g. Let $x \in [a_1, a_n]$ be given, and set $j := \min\{j \in \{1, n-1\} : x \in [a_j, a_{j+1}]\}$. Then

$$\begin{aligned}
\left|f(x) - g(x)\right| &= \left|f(x) - h(p_j(x), g_j(x)) - h(p_{j+1}(x), g_{j+1}(x))\right| \\
&\leq \left|f(x) - p_j(x)g_j(x) - p_{j+1}(x)g_{j+1}(x)\right| \\
&\quad + \left|p_j(x)g_j(x) - h(p_j(x), g_j(x))\right| + \left|p_{j+1}(x)g_{j+1}(x) - h(p_{j+1}(x), g_{j+1}(x))\right| \\
&\leq p_j(x)\left|f(x) - g_j(x)\right| + p_{j+1}(x)\left|f(x) - g_{j+1}(x)\right| + \epsilon + \epsilon \\
&\leq p_j(x)\epsilon + p_{j+1}(x)\epsilon + 2\epsilon.
\end{aligned}$$

□

Proof of Lemma 3.6. Set $c := 1/b$ and $r := \lceil b \ln(1/(\epsilon a))/a \rceil$ and $\epsilon_0 := \epsilon/(r^2 c)$. For $i \in \{0, \ldots, r\}$, let $h_i : [0,1] \to [0,1]$ denote a ReLU network ϵ_0-approximation to the map $x \mapsto x^i$; by Lemma 3.3, h_i has $\mathcal{O}(\ln(1/\epsilon_0)^2$ nodes and layers. Define $q : [0,1] \to \mathbb{R}$ as

$$q(x) := c \sum_{i=0}^{r} h_i(1 - cx).$$

By construction, q is a ReLU network with $\mathcal{O}(r \ln(1/\epsilon_0)^2)$ nodes and $\mathcal{O}(\ln(1/\epsilon_0)^2)$ layers.

For the approximation property of q, let $x \in [a, b]$ be given, and note

$$\begin{aligned}
\left|q(x) - \frac{1}{x}\right| &\leq \left|q(x) - c\sum_{i=0}^{r}(1-cx)^i\right| + \left|c\sum_{i=0}^{r}(1-cx)^i - \frac{1}{x}\right| \\
&\leq c\sum_{i=0}^{r}\left|h_i(1-cx) - (1-cx)^i\right| + \left|c\sum_{i=0}^{r}(1-cx)^i - \frac{c}{1-(1-cx)}\right| \\
&\leq \epsilon + \left|c\sum_{i=0}^{r}(1-cx)^i - c\sum_{i=0}^{\infty}(1-cx)^i\right| \\
&= \epsilon + c\sum_{i=r+1}^{\infty}(1-cx)^i \\
&= \epsilon + \frac{c(1-cx)^{r+1)}}{1-(1-cx)} \\
&\leq \epsilon + \frac{\exp(-cx(r+1))}{x} \\
&\leq \epsilon + \frac{\exp(-car)}{a} \\
&\leq \epsilon + \epsilon.
\end{aligned}$$

□

Proof of Lemma 3.5. Set $\epsilon_0 := \epsilon/3$. For $i \in \{1, \ldots, k\}$, Let $\tilde{q}_i$ denote the ReLU network ϵ_0-approximation to $1/x$ along $[2^{-i}, 2^{-i+1}]$; by Lemma 3.6, $\tilde{q}_i$ has $\mathcal{O}(k^2 \ln(1/\epsilon)^2)$ layers and $\mathcal{O}(k^3 \ln(1/\epsilon)^3)$ nodes. Furthermore, set $q_i := \max\{2^i, \min\{0, \tilde{q}_i\}\}$, which has the same approximation and size properties of $\tilde{q}_i$. Applying Lemma 3.7 with $B := 2^k$ and reals $a_i := 2^{i-k-1}$ for $i \in \{0, \ldots, k+2\}$ and functions $(q_i)_{i=1}^k$, it follows that there exists $q : \mathbb{R} \to \mathbb{R}$ which ϵ-approximates $1/x$ along $[2^{-k}, 1]$ with size $\mathcal{O}(k^2 \ln(1/\epsilon) + k^4 \ln(1/\epsilon)^3)$ and depth $\mathcal{O}(k \ln(1/\epsilon) + k^2 \ln(1/\epsilon)^2)$. □

Putting the pieces together gives the proof of the second part of the main theorem.

Proof of part 1 of Theorem 1.1. Define $\epsilon_0 := \epsilon/2^{2k+3}$, and use Lemmas 3.2, 3.4 and 3.5 to choose ReLU network approximations f_p and f_q to p and q at resolution ϵ_0, as well as ReLU network f for multiplication along $[0,1]^2$ and g to approximate $x \mapsto 1/x$ along $[2^{-k-1}, 1]$, again at resolution ϵ_0. The desired network will compute the function h, defined as

$$h(x) := 2^{k+1} f(f_p(x), 2^{-k-1} g(f_q(x))).$$

Combining the size bounds from the preceding lemmas, h itself has size bound

$$\begin{aligned}&\mathcal{O}\left(\min\left\{sr\ln(sr/\epsilon_0), sd\ln(dsr/\epsilon_0)^2\right\}\right) + \mathcal{O}\left(\ln(1/\epsilon_0)\right) + \mathcal{O}\left(k^4\ln(1/\epsilon_0)^3\right)\\ &= \mathcal{O}\left(\min\left\{srk\ln(sr/\epsilon), sdk^2\ln(dsr/\epsilon)^2\right\} + k^7\ln(1/\epsilon)^3\right).\end{aligned}$$

Before verifying the approximation guarantee upon h, it is necessary to verify that the inputs to f and g are of the correct magnitude, so that Lemmas 3.2 and 3.5 may be applied. Note firstly that $g(f_q(x)) \in [1, 2^{k+1}]$, since $q(x) \in [2^{-k}, 1]$ implies $f_q(x) \in [2^{-k} - \epsilon_0, 1] \subseteq [2^{-k-1}, 1]$. Thus $2^{-k-1}g(f_q(x)) \in [0,1]$, and so both arguments to f within the definition of h are within $[0,1]$. Consequently, the approximation guarantees of Lemmas 3.2, 3.4 and 3.5 all hold, whereby

$$\begin{aligned}h(x) - \frac{p(x)}{q(x)} &= 2^{k+1} f\left(f_p(x), 2^{-k-1}g(f_q(x))\right) - \frac{p(x)}{q(x)}\\ &\leq f_p(x)g(f_q(x)) - \frac{p(x)}{q(x)} + 2^{k+1}\epsilon_0\\ &\leq \frac{f_p(x)}{f_q(x)} - \frac{p(x)}{q(x)} + f_p(x)\epsilon_0 + 2^{k+1}\epsilon_0\\ &\leq \frac{p(x)+\epsilon_0}{q(x)-\epsilon_0} - \frac{p(x)}{q(x)} + f_p(x)\epsilon_0 + 2^{k+1}\epsilon_0\\ &\leq \frac{p(x)q(x) + q(x)\epsilon_0 - p(x)q(x) + p(x)\epsilon_0}{q(x)(q(x)-\epsilon_0)} + f_p(x)\epsilon_0 + 2^{k+1}\epsilon_0\\ &= \frac{\epsilon_0}{q(x)-\epsilon_0} + \frac{p(x)\epsilon_0}{q(x)(q(x)-\epsilon_0)} + f_p(x)\epsilon_0 + 2^{k+1}\epsilon_0\\ &\leq 2^{k+1}\epsilon_0 + 2^{2k+1}\epsilon_0 + f_p(x)\epsilon_0 + 2^{k+1}\epsilon_0\\ &\leq \epsilon.\end{aligned}$$

The proof of the reverse inequality is analogous. $\square$

B.2. Proof of Proposition 1.5

Proof of Proposition 1.5. By (Telgarsky, 2015, Lemma 2.1), a ReLU network g with at most m nodes in each of at most l layers computes a function which is affine along intervals forming a partition of $\mathbb{R}$ of cardinality at most $N' \leq (2m)^l$. Further subdivide this collection of intervals at any point where g intersects $f(x) = 1/x$; since f is convex and g is affine within each existing piece of the subdivision, then the number of intervals is at most three times as large as before. Together, the total number of intervals N'' now satisfies $N'' \leq 3(2m)^l$. Finally, intersect the family of intervals with $[1/2, 3/4]$, obtaining a final number of intervals $N \leq 3(2m)^l$.

Let $(U_1, \ldots, U_N)$ denote this final partition of $[1/2, 3/4]$, and let $(\delta_1, \ldots, \delta_N)$ denote the corresponding interval lengths. Let $S \subseteq \{1, \ldots, N\}$ index the subcollection of intervals with length at least $1/(8N)$, meaning $S := \{j \in \{1, \ldots, N\} : \delta_j \geq 1/(8N)\}$. Then

$$\sum_{j\in S}\delta_j = \frac{1}{4} - \sum_{j\notin S}\delta_j > \frac{1}{4} - \frac{N}{8N} = \frac{1}{8}.$$

Consider now any interval U_j with endpoints $\{a, b\}$. Since $1/2 \leq a < b \leq 3/4$, then f satisfies $128/27 \leq f'' \leq 16$. In order to control the difference between f and g along U_j, consider two cases: either $f \geq g$ along this interval, or $f \leq g$ along this interval (these are the only two cases due to the subdivisions above).

- If $f \geq g$, then g can be taken to be a tangent to f at some point along the interval $[a,b]$ (otherwise, the distance can always be only decreased by moving g up to be a tangent). Consequently, $g(x) := f(c) + f'(c)(x-c)$ for some $c \in [a,b]$, and by convexity and since $f'' \geq 128/27$ over this interval,

$$\begin{aligned}
\int_a^b |f(x)-g(x)|\,\mathrm{d}x &\geq \min_{c\in[a,b]} \int_a^b \Big((f(c)+f'(c)(x-c)+f''(b)(x-c)^2/2) - (f(c)+f'(c)(x-c)) \Big)\,\mathrm{d}x \\
&= \min_{c\in[a,b]} \int_a^b f''(b)(x-c)^2/2\,\mathrm{d}x \\
&\geq \frac{64}{27} \min_{c\in[a,b]} \left(\frac{(b-c)^3-(a-c)^3}{3} \right). \\
&= \frac{64}{81} \min_{\alpha\in[0,1]} \Big((\alpha(b-a))^3 + ((1-\alpha)(b-a))^3 \Big) \\
&= \frac{16(b-a)^3}{81}.
\end{aligned}$$

- On the other hand, if $g \geq f$, then g passes above the secant line h between $(a, f(a))$ and $(b, f(b))$. The area between f and g is at least the area between f and h, and this latter area is bounded above by a triangle of width $(b-a)$ and height

$$\begin{aligned}
\frac{f(a)+f(b)}{2} - f\big((a+b)/2\big) &= \frac{1}{2}\left(\frac{1}{a}+\frac{1}{b}-\frac{1}{a+b}\right) \\
&= \frac{1}{2ab(a+b)}\big(b(a+b)+a(a+b)-ab\big) \\
&\geq \frac{1}{2ab(a+b)}\big(b(a+b)+a(a+b)-ab\big) \\
&\geq \frac{3/2}{4}.
\end{aligned}$$

  Combining this with $b-a \leq 1/4$, the triangle has area at least $3(b-a)/16 \geq 3(b-a)^3$.

Combining these two cases and summing across the intervals of S (where $j \in S_j$ implies $\delta_j \geq 1/(8N)$),

$$\begin{aligned}
\int_{[1/2,3/4]} |f(x)-g(x)|\,\mathrm{d}x &\geq \sum_{j\in S} \int_{U_j} |f(x)-g(x)|\,\mathrm{d}x \\
&\geq \sum_{j\in S} \frac{\delta_j^3}{6} \\
&\geq \frac{1}{6(8N)^2} \sum_{j\in S} \delta_j \\
&\geq \frac{1}{27648(2m)^{2l}}.
\end{aligned}$$

If $m < (27648\epsilon)^{-1/(2l)}/2$, then

$$\int_{[1/2,3/4]} |f(x)-g(x)|\,\mathrm{d}x \geq \frac{1}{27648(2m)^{2l}} > \epsilon.$$

$\square$