Expressiveness of Rectifier Networks

Xingyuan Pan XPAN@CS.UTAH.EDU
Vivek Srikumar SVIVEK@CS.UTAH.EDU
The University of Utah, Salt Lake City, UT 84112, USA

Abstract

Rectified Linear Units (ReLUs) have been shown to ameliorate the vanishing gradient problem, allow for efficient backpropagation, and empirically promote sparsity in the learned parameters. They have led to state-of-the-art results in a variety of applications. However, unlike threshold and sigmoid networks, ReLU networks are less explored from the perspective of their expressiveness. This paper studies the expressiveness of ReLU networks. We characterize the decision boundary of two-layer ReLU networks by constructing functionally equivalent threshold networks. We show that while the decision boundary of a two-layer ReLU network can be captured by a threshold network, the latter may require an exponentially larger number of hidden units. We also formulate sufficient conditions for a corresponding logarithmic reduction in the number of hidden units to represent a sign network as a ReLU network. Finally, we experimentally compare threshold networks and their much smaller ReLU counterparts with respect to their ability to learn from synthetically generated data.

1. Introduction

A neural network is characterized by its architecture, the choices of activation functions, and its parameters. We see several activation functions in the literature – the most common ones being threshold, logistic, hyperbolic tangent and rectified linear units (ReLUs). In recent years, deep neural networks with rectifying neurons – defined as $R(x) = \max(0, x)$ – have shown state-of-the-art performance in several tasks such as image and speech classification (Glorot et al., 2011; Nair & Hinton, 2010; Krizhevsky et al., 2012; Maas et al., 2013; Zeiler et al., 2013).

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

ReLUs possess several attractive computational properties. First, compared to deep networks with sigmoidal activation units, ReLU networks are less affected by the vanishing gradient problem (Bengio et al., 1994; Hochreiter, 1998; Glorot et al., 2011). Second, rectifying neurons encourage sparsity in the hidden layers (Glorot et al., 2011). Third, gradient back propagation is efficient because of the piecewise linear nature of the function. For example, Krizhevsky et al. (2012) report that a convolutional neural network with ReLUs is six times faster than an equivalent one with hyperbolic tangent neurons. Finally, they have been empirically been shown to generalize very well.

Despite these clear computational and empirical advantages, the expressiveness of rectifier units is less studied unlike sigmoid and threshold units. In this paper, we address the following question: *Which Boolean functions do ReLU networks express?* We analyze the expressiveness of shallow ReLU networks by characterizing their equivalent threshold networks. The goal of our analysis is to offer a formal understanding for the successes of ReLUs by comparing them to threshold functions that are well studied (e.g. Hajnal et al., 1993). To this end, the contributions of this paper are:

1. We provide a constructive proof that two layer ReLU networks are equivalent to exponentially larger threshold networks. Furthermore, we show that there exist two layer ReLU networks that *cannot* be represented by any smaller threshold networks.
2. We use this characterization to define a sufficient condition for compressing an arbitrary threshold network into a logarithmically smaller ReLU network.
3. We identify a relaxation of this condition that is applicable if we treat hidden layer predictions as a multiclass classification, thus requiring equivalence of hidden layer states instead of the output state.

1.1. Expressiveness of Networks: Related Work

From the learning point of view, the choice of an activation function is driven by two related aspects: the expres-

siveness of a given network using the activation function, and the computational complexity of learning. Though this work studies the former, we briefly summarize prior work along both these lines.

Any continuous function can be approximated to arbitrary accuracy with only one hidden layer of sigmoid units (Cybenko, 1989), leading to neural networks being called "universal approximators". With two layers, even discontinuous functions can be represented. Moreover, the approximation error (for real-valued outputs) is insensitive to the choice of activation functions from among several commonly used ones (DasGupta & Schnitger, 1993), provided we allow the size of the network to increase polynomially and the number of layers to increase by a constant factor. Similarly, two layer threshold networks are capable of representing any Boolean function. However, these are existence statements; for a general target function, the number of hidden units may be exponential in the input dimensionality. Maass et al. (1991; 1994) compare sigmoid networks with threshold networks and point out that the former can be more expressive than similar-sized threshold networks.

There has been some recent work that looks at the expressiveness of feed-forward ReLU networks. Because the rectifier function is piece-wise linear, any network using only ReLUs can only represent piece-wise linear functions. Thus, the number of linear partitions of input space by the network can be viewed as a measure of its expressiveness. Pascanu et al. (2014) and Montufar et al. (2014) show that for the same number of ReLUs, a deep architecture can represent functions with exponentially more linear regions than a shallow architecture. While more linear regions indicate that more complex functions can be represented, it does not directly tell us how expressive a function is; at prediction time, we cannot directly correlate the number of regions to the way we make the prediction. Another way of measuring the expressiveness of a feed-forward networks is by considering its classification error; Telgarsky (2015) compares shallow and deep ReLU networks in this manner.

The learning complexity of neural networks using various activation functions has also been studied. For inputs from the Boolean hypercube, the two-layer networks with threshold activation functions is not efficiently learnable (e.g. Blum & Rivest, 1992; Klivans & Sherstov, 2006; Daniely et al., 2014). Without restricting the weights, two layer networks with sigmoid or ReLU activations are also not efficiently learnable. We also refer the reader to Livni et al. (2014) that summarizes and describes positive and negative learnability results for various activations.

2. What do ReLUs express?

To simplify our analysis, we primarily focus on shallow networks with one hidden layer with n units and a single binary output. In all cases, the hidden layer neurons are the object of study. The output activation function is always the threshold function. In the rest of the paper, we use bold-faced letters to denote vectors. Input feature vectors and output binary labels are represented by $\mathbf{x}$ and $y \in \{\pm 1\}$ respectively. The number of hidden units is n. The weights and bias for the k^{th} rectifier are $\mathbf{u}_k$ and b_k; the weights and bias for the k^{th} sign units are $\mathbf{v}_k$ and d_k. The weights for the output unit are w_1 through w_n, and its the bias is w_0.

2.1. Threshold networks

Before coming to the main results, we will first review the expressiveness of threshold networks. Assuming there are n hidden units and one output unit, the output of the network can be written as

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{n} w_k \operatorname{sgn}\left(\mathbf{v}_k \cdot \mathbf{x} + d_k\right)\right). \qquad (1)$$

Here, both hidden and output activations are the sign function, which is defined as $\operatorname{sgn}(x) = 1$ if $x \geq 0$, and -1 otherwise. Each hidden unit represents one hyperplane (parameterized by $\mathbf{v}_k$ and d_k) that bisects the input space into two half spaces. By choosing different weights in the hidden layer we can obtain arbitrary arrangement of n hyperplanes. The theory of hyperplane arrangement (Zaslavsky, 1975) tells us that for a general arrangement of n hyperplanes in d dimensions, the space is divided into $\sum_{s=0}^{d} \binom{n}{s}$ regions. The output unit computes a linear combination of the hidden output (using the w's) and thresholds it. Thus, for various values of the w's, threshold networks can express intersections and unions of those regions. Figure 1 shows an example of the decision boundary of a two-layer network with three threshold units in the hidden layer.

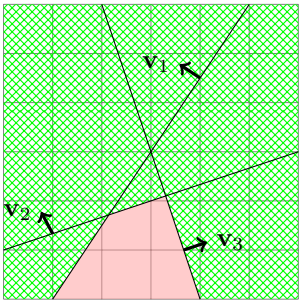

Figure 1. An example of the decision boundary of a two-layer network in two dimensions, with three threshold units in the hidden layer. The arrows point towards the half-space that is classified as positive (the green checked region).

2.2. Rectifier networks

In this section, we will show that the decision boundary of every two-layer neural network with rectifier activations can be represented using a network with threshold activations with two or three layers. However, the number of hidden threshold units can be exponential compared to the number of hidden rectifier units.

Consider a network with one hidden layer of n ReLUs, denoted by $R(\cdot)$. For a d dimensional input $\mathbf{x}$, the output y is computed as

$$y = \text{sgn}\left(w_0 + \sum_{k=1}^{n} w_k R\left(\mathbf{u}_k \cdot \mathbf{x} + b_k\right)\right). \quad (2)$$

Here $\mathbf{u}_k$ and b_k are weight and bias parameters for the ReLUs in the hidden layer, and the w_k's parameterize the output unit. To simplify notation, we will use a_k to denote the the pre-activation input of the k^{th} hidden unit. That is, $a_k(\mathbf{x}) = \mathbf{u}_k \cdot \mathbf{x} + b_k$. This allows us to simplify the output as $\text{sgn}\left(w_0 + \sum_{k\in[n]} w_k R\left(a_k(\mathbf{x})\right)\right)$. Here, $[n]$ is the set of positive integers not more than n. Note that even when not explicitly mentioned, each a_k depends on the $\mathbf{u}_k$ and the b_k parameters.

By definition of the rectifier, for any real number c, we have $cR(x) = \text{sgn}(c)R(|c|x)$. Thus, we can absorb $|w_k|$ into the rectifier function in Eq. (2) without losing generality. That is, other than w_0, all the other output layer weights are only relevant up to sign because their magnitude can be absorbed into hidden layer weights. We can partition the hidden units into two sets $\mathcal{P}$ and $\mathcal{N}$, depending on the sign of the corresponding w_k. That is, let $\mathcal{P} = \{k : k \in [n] \text{ and } w_k = +1\}$ and let $\mathcal{N} = \{k : k \in [n] \text{ and } w_k = -1\}$. We will refer to these partitions as the *positive* and *negative* hidden units respectively.

This observation lets us state the general form of two-layer ReLU networks as:

$$y = \text{sgn}\left(w_0 + \sum_{k\in\mathcal{P}} R\left(a_k(\mathbf{x})\right) - \sum_{k\in\mathcal{N}} R\left(a_k(\mathbf{x})\right)\right). \quad (3)$$

The following two layer rectifier network will serve as our running example through the paper:

$$y = \text{sgn}\left(w_0 + R\left(a_1(\mathbf{x})\right) - R\left(a_2(\mathbf{x})\right) - R\left(a_3(\mathbf{x})\right)\right). \quad (4)$$

This network consists of three ReLUs in the hidden layer, one of which positively affects the pre-activation output and the other two decrease it. Hence, the set $\mathcal{P} = \{1\}$ and the set $\mathcal{N} = \{2, 3\}$.

Using the general representation of a two layer network with rectifier hidden units (Eq. (3)), we can now state our main theorem that analyzes the decision boundary of rectifier networks.

Theorem 1 (Main Theorem)**.** *Consider a two-layer rectifier network with n hidden units represented in its general form (Eq. (3)). Then, for any input $\mathbf{x}$, the following conditions are equivalent:*

1. *The network classifies the example $\mathbf{x}$ as positive.*
2. *There exists a subset $\mathcal{S}_1$ of $\mathcal{P}$ such that, for every subset $\mathcal{S}_2$ of $\mathcal{N}$, we have $w_0 + \sum_{k\in\mathcal{S}_1} a_k(\mathbf{x}) - \sum_{k\in\mathcal{S}_2} a_k(\mathbf{x}) \geq 0$.*
3. *For every subset $\mathcal{S}_2$ of $\mathcal{N}$, there exists a subset $\mathcal{S}_1$ of $\mathcal{P}$ such that $w_0 + \sum_{k\in\mathcal{S}_1} a_k(\mathbf{x}) - \sum_{k\in\mathcal{S}_2} a_k(\mathbf{x}) \geq 0$.*

Before discussing the implications of the theorem, let us see how it applies to our running example in Eq. (4). In this example, $\mathcal{P}$ has two subsets: $\emptyset$ and $\{1\}$, and $\mathcal{N}$ has four subsets: $\emptyset$, $\{2\}$, $\{3\}$ and $\{2, 3\}$. The first and second conditions of Theorem 1 indicate that the prediction is positive if, and only if, at least one of the sets of conditions in Figure 2 hold in entirety.

Each big left brace indicates a system of inequalities all of which should hold; thus essentially the conjunction of the individual inequalities contained within it. We can interpret of the subsets of $\mathcal{P}$ as certificates. In order for the output of Eq. (4) to be positive, we need at least one certificate $\mathcal{S}_1$ (one subset of $\mathcal{P}$) such that for every subset $\mathcal{S}_2$ of $\mathcal{N}$, $w_0 + \sum_{k\in\mathcal{S}_1} a_k(\mathbf{x}) - \sum_{k\in\mathcal{S}_2} a_k(\mathbf{x}) \geq 0$. The two sets of inequalities show the choices of subsets of $\mathcal{N}$ for each of the two possible choices of $\mathcal{S}_1$ (i.e. either $\emptyset$ or $\{1\}$). The above conditions represent a disjunction of conjunctions.

Similarly, employing the first and third conditions of the theorem to our running example gives us:

$$\begin{cases} w_0 \geq 0, & \text{or } w_0 + a_1(\mathbf{x}) \geq 0 \\ w_0 - a_2(\mathbf{x}) \geq 0, & \text{or } w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) \geq 0 \\ w_0 - a_3(\mathbf{x}) \geq 0, & \text{or } w_0 + a_1(\mathbf{x}) - a_3(\mathbf{x}) \geq 0 \\ w_0 - a_2(\mathbf{x}) - a_3(\mathbf{x}) \geq 0, & \text{or } w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) \\ & \quad -a_3(\mathbf{x}) \geq 0 \end{cases} \quad (5)$$

Note that unlike the previous case, this gives us a condition that is a conjunction of disjunctions.

The complete proof of Theorem 1 is given in the supplementary material; here we give a sketch. To prove that condition 1 implies condition 2 we construct a specific subset $\mathcal{S}_1^*$ of $\mathcal{P}$, where $\mathcal{S}_1^* = \{k : k \in \mathcal{P} \text{ and } a_k(\mathbf{x}) \geq 0\}$ and this $\mathcal{S}_1^*$ has the desired property. To prove condition 2 implies condition 1 we make use of a specific subset $\mathcal{S}_2^*$ of $\mathcal{N}$, where $\mathcal{S}_2^* = \{k : k \in \mathcal{N} \text{ and } a_k(\mathbf{x}) \geq 0\}$, to show the example $\mathbf{x}$ has a positive label. That condition 1 implies condition 3 is a direct result of condition 1 implying condition 2. Finally, to prove condition 3 implies condition 1 we use the same $\mathcal{S}_2^*$ again to show $\mathbf{x}$ has a positive label.

$$\begin{cases} w_0 \geq 0, & (\text{with } \mathcal{S}_1 = \emptyset, \mathcal{S}_2 = \emptyset) \\ w_0 - a_2(\mathbf{x}) \geq 0, & (\text{with } \mathcal{S}_1 = \emptyset, \mathcal{S}_2 = \{2\}) \\ w_0 - a_3(\mathbf{x}) \geq 0, & (\text{with } \mathcal{S}_1 = \emptyset, \mathcal{S}_2 = \{3\}) \\ w_0 - a_2(\mathbf{x}) - a_3(\mathbf{x}) \geq 0. & (\text{with } \mathcal{S}_1 = \emptyset, \mathcal{S}_2 = \{2,3\}) \end{cases} \text{(or)} \begin{cases} w_0 + a_1(\mathbf{x}) \geq 0, & (\text{with } \mathcal{S}_1 = \{1\}, \mathcal{S}_2 = \emptyset) \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) \geq 0, & (\text{with } \mathcal{S}_1 = \{1\}, \mathcal{S}_2 = \{2\}) \\ w_0 + a_1(\mathbf{x}) - a_3(\mathbf{x}) \geq 0, & (\text{with } \mathcal{S}_1 = \{1\}, \mathcal{S}_2 = \{3\}) \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) - a_3(\mathbf{x}) \geq 0. & (\text{with } \mathcal{S}_1 = \{1\}, \mathcal{S}_2 = \{2,3\}) \end{cases}$$

Figure 2. The sets of inequalities that should hold for the running example to predict an input as positive.

Discussion. The only difference between the second and the third conditions of the theorem is the order of the universal and existential quantifiers over the positive and negative hidden units, $\mathcal{P}$ and $\mathcal{N}$ respectively. More importantly, in both cases, the inequality condition over the subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ is identical. Normally, swapping the order of the quantifiers does not give us an equivalent statement; but here, we see that doing so retains meaning because, in both cases, the output is positive for the corresponding input.

For any subsets $\mathcal{S}_1 \subseteq \mathcal{P}$ and $\mathcal{S}_2 \subseteq \mathcal{N}$, we can write the inequality condition as a Boolean function $B_{\mathcal{S}_1,\mathcal{S}_2}$:

$$B_{\mathcal{S}_1,\mathcal{S}_2}(\mathbf{x}) = \begin{cases} \text{true}, & w_0 + \sum_{k\in\mathcal{S}_1} a_k(\mathbf{x}) \\ & - \sum_{k\in\mathcal{S}_2} a_k(\mathbf{x}) \geq 0 \\ \text{false}, & w_0 + \sum_{k\in\mathcal{S}_1} a_k(\mathbf{x}) \\ & - \sum_{k\in\mathcal{S}_2} a_k(\mathbf{x}) < 0 \end{cases} \tag{6}$$

If the sizes of the positive and negative subsets are n_1 and n_2 respectively (i.e, $n_1 = |\mathcal{P}|$ and $n_2 = |\mathcal{N}|$), then we know that $\mathcal{P}$ has 2^{n_1} subsets and $\mathcal{N}$ has 2^{n_2} subsets. Thus, there are $2^{n_1+n_2}$ such Boolean functions. Then, by virtue of conditions 1 and 2 of theorem 1, we have[1]

$$y = \vee_{\mathcal{S}_1\subseteq\mathcal{P}} \left[\wedge_{\mathcal{S}_2\subseteq\mathcal{N}} B_{\mathcal{S}_1,\mathcal{S}_2}(\mathbf{x})\right],$$

where $\wedge_{\mathcal{S}_2}$ indicates a conjunction over all different subsets $\mathcal{S}_2$ of $\mathcal{N}$, and $\vee_{\mathcal{S}_1}$ indicates a disjunction over all different subsets $\mathcal{S}_1$ of $\mathcal{P}$. This expression is in the disjunctive normal form (DNF), where each conjunct contains 2^{n_2} B's and there are 2^{n_1} such terms. Since each B simplifies into a hyperplane in the input space, this characterizes the decision boundary of the ReLU network as a DNF expression over these hyperplane decisions.

Similarly, by conditions 1 and 3, we have $y = \wedge_{\mathcal{S}_2}\left[\vee_{\mathcal{S}_1} B_{\mathcal{S}_1,\mathcal{S}_2}(\mathbf{x})\right]$. This is in the conjunctive normal form (CNF), where each disjunctive clause contains 2^{n_1} Boolean values and there are 2^{n_2} such clauses.

An corollary is that if the hidden units of the ReLU network are all positive (or negative), then the equivalent threshold network is a pure disjunction (or conjunction).

[1] We write y as a Boolean with $y = 1$ and $y = -1$ representing true and false respectively.

3. Comparing ReLU and threshold networks

In the previous section, we saw that ReLU networks can express Boolean functions that correspond to much larger threshold networks. Of course, threshold activations are not generally used in applications; nonetheless, they are well understood theoretically *and* they emerge naturally as a result of the analysis above. Using threshold functions as a vehicle to represent the decision boundaries of ReLU networks, naturally leads to two related questions that we will address in this section. First, given an arbitrary ReLU network, can we construct an equivalent threshold network? Second, given an arbitrary threshold network, how can we represent it using ReLU network?

3.1. Converting from ReLU to Threshold

Theorem 1 essentially gives us a constructive way to represent an arbitrary two layer ReLU network given in Eq. (3) as a three-layer threshold network. For every choice of the subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ of the positive and negative units, we can define a Boolean function $B_{\mathcal{S}_1,\mathcal{S}_2}$ as per Eq. (6). By definition, each of these is a threshold unit, giving us $2^{n_1+n_2}$ threshold units in all. (Recall that n_1 and n_2 are the sizes of $\mathcal{P}$ and $\mathcal{N}$ respectively.) Since the decision function is a CNF or a DNF over these functions, it can be represented using a two-layer network over the B's, giving us three layers in all. We put all $2^{n_1+n_2}$ threshold units in the first hidden layer, separated into 2^{n_1} groups, with each group comprising of 2^{n_2} units.

Figure 3 shows the threshold network corresponding to our running example from Eq. (4). For brevity, we use the notation $B_{i,j}$ to represent the first hidden layer, with i and j indexing over the subsets of $\mathcal{P}$ and $\mathcal{N}$ respectively. The B's can be grouped into two groups, with units in each group sharing the same subset $\mathcal{S}_1$ but with different $\mathcal{S}_2$. Note that, these nodes are linear threshold units corresponding to the inequalities in in Fig. 2. In the second hidden layer, we have one threshold unit connected to each group of units in the layer below. The weight for each connection unit is 1 and the bias is $2^{n_2} - 1$, effectively giving us a conjunction of the previous layer nodes. The second hidden layer has 2^{n_1} such units. Finally, we have one unit in the output layer, with all weights being 1 and bias being $1 - 2^{n_1}$, simulating a disjunction of the decisions of the layer below. As discussed in the previous section, we can also construct a

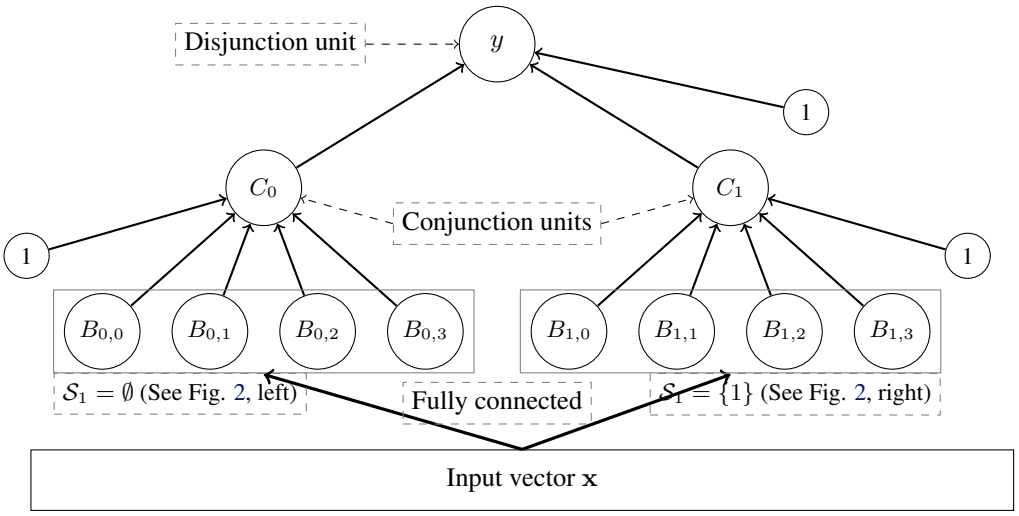

Figure 3. A threshold network corresponding to the running example. The dotted boxes label the various components of the network. See the text above for details.

threshold networks using CNFs, with $2^{n_1+n_2}$ units in the first hidden layer, and 2^{n_2} units in the second hidden layer.

3.2. Boolean Expressiveness of ReLU Networks

Our main theorem shows that for an arbitrary two-layer ReLU network, we *can always* represent it using a three-layer threshold network in which the number of threshold units is exponentially more than the number of ReLUs. However, this does not imply that actually need that many thresholds units. A natural question to ask is: can we represent the same ReLU network without the expense of exponential number of threshold units? In general, the answer is no, because there are families of ReLU network that need at least an exponential number of threshold units to represent.

We can formalize this for the case of ReLU networks where the hidden nodes are either all positive or negative – that is, either $\mathcal{P}$ or $\mathcal{N}$ is the empty set. We restrict ourselves to this set because we can represent such networks using a two-layer threshold network rather than the three-layer one in the previous section. We will consider the set of all Boolean functions in d dimensions expressed by such a ReLU network with n hidden units. Let $\Gamma_R(n, d)$ represent this set. Similarly, let $\Gamma_T(n, d)$ denote the set of such functions expressed by threshold networks with n hidden units. We can summarize the Boolean expressiveness of ReLU networks via the following two-part theorem:

Theorem 2. *For rectifier networks with $n > 1$ hidden units, all of which are either positive or negative, and for all input dimensionalities $d \geq n$, we have*

1. $\Gamma_R(n, d) \subseteq \Gamma_T(2^n - 1, d)$*, and,*
2. $\Gamma_R(n, d) \not\subset \Gamma_T(2^n - 2, d)$.

Before seeing the proof sketch, let us look at an intuitive explanation of this result. This theorem tells us that the upper bound on the number of threshold units corresponding to the ReLU network is a tight one. In other words, not only can ReLU networks express Boolean functions that correspond to much larger threshold networks, there are some ReLU networks that can *only* be expressed by such large networks! This theorem may give us some intuition into the successes of ReLU networks.

Note, however, that this theorem does not resolve the question of what fraction of all ReLU networks can be expressed using fewer than exponential number of hidden threshold units. Indeed, if the inputs were only Boolean and the output was allowed to be real-valued, then Martens et al. (2013, theorem 6) show that a two layer ReLU network can be simulated using only a quadratic number of threshold units. (In contrast, Theorem 2 above considers the case of real-valued inputs, but Boolean outputs.)

The proof is based on Theorem 1. From Theorem 1, we know the decision boundary of the rectifier network with n positive ReLUs (with $\mathcal{N}$ being the empty set) can be determined from $2^n - 1$ hyperplanes. In other words, the region of the input space that is classified as negative is the polytope that is defined by the intersection of these $2^n - 1$ half-planes. We can show by construction that there are rectifier networks for which the negative region is a polytope defined by $2^n - 1$ bounding surfaces, thus necessitating *each* of the $2^n - 1$ half-planes as predicted by the main theorem, irrespective of how the corresponding threshold network is constructed. In other words, the number of threshold units cannot be reduced. The proof for the case of all negative ReLUs is similar. The complete proof of the theorem is given in the supplementary material.

3.3. Converting Thresholds to ReLUs

So far, we have looked at threshold networks corresponding to ReLU networks. The next question we want to answer is under what condition we can use ReLUs to represent the same decision boundary as a threshold network. In this section, we show a series of results that address various facets of this question. The longer proofs are in the appendices.

First, with no restrictions in the number of ReLUs in the hidden layer, then we can always construct a rectifier network that is equivalent to a threshold network. In fact, we have the following lemma:

Lemma 1. *Any threshold network with n units can be approximated to arbitrary accuracy by a rectifier network with $2n$ units.*

Proof. Consider a threshold unit with weight vector $\mathbf{v}$ and bias d, we have

$$\text{sgn}(\mathbf{v}\cdot\mathbf{x}+d) \simeq \frac{1}{\epsilon}\big[R(\mathbf{v}\cdot\mathbf{x}+d+\epsilon)-R(\mathbf{v}\cdot\mathbf{x}+d-\epsilon)\big]-1.$$

where ϵ is an arbitrary small number which determines the approximation accuracy. □

This result is akin to the simulation results from (Maass et al., 1994) that compares threshold and sigmoid networks.

Given the exponential increase in the size of the threshold network to represent a ReLU network (Theorem 2), a natural question is whether we can use only logarithmic number of ReLUs to represent any arbitrary threshold network. In the general case, the following lemma points out that this is not possible.

Lemma 2. *There exists a two-layer network with n hidden threshold units for which it is not possible to construct an equivalent two-layer ReLU network with fewer number of hidden units.*

We provide such an example with n hidden threshold units in the supplementary material. This lemma, in conjunction with Theorem 2 effectively points out that by employing a rectifier network, we are exploring a *subset* of much larger threshold networks.

Furthermore, despite the negative result of the lemma, in the general case, we can identify certain specific threshold networks that can be compressed into logarithmically smaller ones using ReLUs. Suppose we wish to compress a two layer threshold network with three sign hidden units into a ReLU network with $\lceil \log_2 3 \rceil = 2$ hidden units. The sign network can be represented by

$$y = \text{sgn}(2 + \text{sgn}\,(\mathbf{v}_1\cdot\mathbf{x}+d_1) + \text{sgn}\,(\mathbf{v}_2\cdot\mathbf{x}+d_2) + \text{sgn}\,(\mathbf{v}_3\cdot\mathbf{x}+d_3))$$

Suppose one of the weight vectors can be written as the linear combination of the other two but its bias can not. That is, for some p and q, if $\mathbf{v}_3 = p\mathbf{v}_1 + q\mathbf{v}_2$ and $d_3 \neq pd_1 + qd_2$. Then, we can construct the following equivalent ReLU network that is equivalent:

$$y = \text{sgn}\,(-1 + R\,(\mathbf{u}_1\cdot\mathbf{x}+b_1) + R\,(\mathbf{u}_2\cdot\mathbf{x}+b_2))\,,$$

where
$$r = \frac{1}{d_3 - pd_1 - qd_2},$$
$$\mathbf{u}_1 = pr\mathbf{v}_1,$$
$$\mathbf{u}_2 = qr\mathbf{v}_2,$$
$$b_1 = prd_1 + 1,$$
$$b_2 = qrd_2 + 1.$$

This equivalence can be proved by applying Theorem 1 to the constructed ReLU network. It shows that in two dimensions, we can use two ReLUs to represent three linearly independent sign units.

We can generalize this result to the case of a two-layer threshold network with 2^n hidden threshold units that represents a disjunction over the hidden units. The goal is to find that under what condition we can use only n rectifier units to represent the same decision. To do so, we will use binary encoding matrix T_n of size $n \times 2^n$ whose i^{th} column is the binary representation of $i-1$. For example, the binary encoding matrix for $n = 3$ is given by T_3,

$$T_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Lemma 3. *Consider a two-layer threshold network with 2^n threshold units in the hidden layer whose output represents a disjunction over the hidden units, i.e., the final output is positive if and only if at least one of the hidden-unit outputs is positive. That is,*

$$y = \text{sgn}\left(2^n - 1 + \sum_{k=1}^{2^n} \text{sgn}\,(\mathbf{v}_k\cdot\mathbf{x}+d_k)\right). \quad (7)$$

This decision can be represented using a two-layer rectifier network with n hidden units, if the weight parameters of the threshold units can be factored in the following form:

$$\begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_{2^n} \\ d_1 & \cdots & d_{2^n} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_n & \mathbf{0} \\ b_1 & \cdots & b_n & w_0 \end{bmatrix} \begin{bmatrix} T_n \\ \mathbf{e}_{2^n} \end{bmatrix} \quad (8)$$

where $\mathbf{e}_{2^n}$ is a 2^n dimensional row vector of all ones and $\mathbf{0}$ is a vector of all zeros.

Proof. If the weight parameters $\mathbf{v}_k$ and d_k can be written in the form as in Eq. (8), then we can construct the two-layer rectifier network,

$$y = \text{sgn}\big[w_0 + \sum_{k=1}^{n} R(\mathbf{u}_k\cdot\mathbf{x}+b_k)\big]. \quad (9)$$

Then by virtue of theorem 1, the decision boundary of the rectifier network in Eq. (9) is the same as the decision boundary of the threshold network in Eq. (7). □

Note that this lemma only identifies sufficient conditions for the logarithmic reduction in network size. Identifying both necessary and sufficient conditions for such a reduction is an open question.

4. Hidden Layer Equivalence

Lemma 3 studies a specific threshold network, where the output layer is a disjunction over the hidden layer units. For this network, we can define an different notion of equivalence between networks by studying the hidden layer activations. We do so by interpreting the hidden layer state of the network as a specific kind of a multiclass classifier that either rejects inputs or labels them. If the output is negative, then clearly none of the hidden layer units are active and the input is rejected. If the output layer is positive, then at least one of the hidden layer units is active and the multiclass label is given by the maximum scoring hidden unit, namely $\arg\max_k \mathbf{v}_k \cdot \mathbf{x} + d_k$.

For threshold networks, the number of hidden units is equal to the number of classes. The goal is to learn the same concept with rectifier units, hopefully with fewer rectifier units than the number of classes. Suppose a ReLU network has n hidden units, then its hidden layer prediction is the highest scoring hidden unit of the corresponding threshold network that has 2^n hidden units. We now define *hidden layer equivalence* of two networks as follows: A threshold network and a ReLU network are equivalent if both their hidden layer predictions are identical.

We already know from lemma 3 that if the weight parameters of the true concept satisfy Eq. (8), then instead of learning 2^n threshold units we can just learn n rectifier units. For simplicity, we write Eq. (8) as $V = UT$ where

$$V = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_{2^n} \\ d_1 & \cdots & d_{2^n} \end{bmatrix} \quad U = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_n & \mathbf{0} \\ b_1 & \cdots & b_n & w_0 \end{bmatrix}$$

and

$$T = \begin{bmatrix} T_n \\ \mathbf{e}_{2^n} \end{bmatrix}$$

For simplicity of notation, we will assume that the input features $\mathbf{x}$ includes a constant bias feature in the last position. Thus, the vector $V^T\mathbf{x}$ represents the pre-activation score for each class.

Now, we consider threshold networks with parameters such that there is *no* ReLU (defined by the matrix U) that satisfies this condition. Instead, we find a rectifier network with parameters U that satisfies the following condition:

$$U = \text{argmin}_U \|(V - UT)^T\|_\infty, \tag{10}$$

Here $\|\cdot\|_\infty$ is the induced infinity norm, defined for any matrix A as $\|A\|_\infty = \sup_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty}$.

If we have a matrix U such that V and UT are close in the sense of induced infinity norm, then we have the following about their equivalence.

Theorem 3. *If the true concept of a 2^n-class classifier is given by a two-level threshold network in Eq. (7), then we can learn a two-layer rectifier network with only n hidden units of the form in Eq. (9) that is hidden layer equivalent to it, if for any example $\mathbf{x}$, we have*

$$\|(V - UT)^T\|_\infty \leq \frac{\gamma(\mathbf{x})}{2\|\mathbf{x}\|_\infty}, \tag{11}$$

where $\gamma(\mathbf{x})$ is the multiclass margin for $\mathbf{x}$, defined as the difference between its highest score and second-highest scoring classes.

The proof, in the supplementary material, is based on the intuition that for hidden layer equivalence, as defined above, *only* requires that the highest scoring label needs to be the same in the two networks rather than the actual values of the scores. If V and UT are closed in the sense of induced infinity norm, then the highest scoring hidden unit will be invariant regardless of which network is used.

5. Experiments

We have seen that every two-layer rectifier network expresses the decision boundary of a three-layer threshold network. If the output weights of the former are all positive, then a two-layer threshold network is sufficient. (See the discussion in §3.2.) However, the fact that rectifier network can express the same decision boundary more compactly does not guarantee learnability because of optimization issues. Specifically, in this section, we study the following question using synthetic data: *Given a rectifier network and a threshold network with same decision boundary, can we learn one using the data generated from another using backpropagation?*

5.1. Data generation

We use randomly constructed two-layer rectifier networks to generate labeled examples. To do so, we specify various values of the input dimensionality and the number of hidden ReLU units in the network. Once we have the network, we randomly generate the input points and label them using the network. Using generated data we try to recover both the rectifier network and the threshold network, with varying number of hidden units. We considered input dimensionalities 3, 10 and 50 and in each case, used 3 or 10 hidden units. This gave us six networks in all. For each network, we generated 10000 examples and 1500 of which are used as test examples.

5.2. Results and Analysis

For each dataset, we compare three different network architectures. The key parameter that varies across datasets is n, the number of hidden ReLU units in the network that generated the data. The first setting learns using a ReLU network with n hidden units. The second setting uses the activation function $\tanh(cx)$, which we call the compressed $\tanh$ activation. For large values of c, this effectively simulates the threshold function. In the second setting, the number of hidden units is still n. The final setting learns using the compressed $\tanh$, but with 2^n hidden units following §2.2.

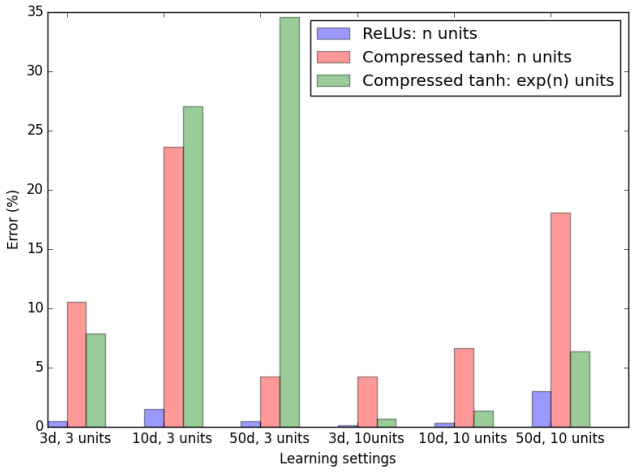

Figure 4. Test error for different learning settings. The x-axis specifies the number of ReLUs used for data generalization and the dimensionality. Each dataset is learned using ReLU and compressed $\tanh$ activations with different number of hidden units. Learning rate is selected with cross-validation from $\{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. L_2-regularization coefficient is 10^{-4}. We use early-stopping optimization with a maximum of 1000 epochs. The minibatch size is 20. For the compressed $\tanh$, we set $c = 10000$.

Figure 4 shows the results on the six datasets. These results verify several aspects of our theory. First, learning using ReLUs always succeeds with low error (purple bars, left). This is expected – we know that our hypothesis class can express the true concept and training using backpropagation can successfully find it. Second, learning using compressed $\tanh$ with same number of units cannot recover the true concept (red bars, middle). This performance drop is as expected, since compressed $\tanh$ is just like sign activation, and we know in this case we need exponentially more hidden units.

Finally, the performances of learning using exponential number of compressed $\tanh$ (green bars, right) are not always good.[2] In this case, from the analysis in §2, we know the hypothesis can certainly express the true concept; yet learning does not always succeed! In fact, for the first three groups, where we have three ReLUs for data generation, the error for the learned classifier is rather large, suggesting that even though the true concept can be expressed, it is not found by backpropagation. For the last three groups, where we have 10 hidden ReLUs for data generation, using exponential number of compressed $\tanh$ does achieve better performance. We posit that this incongruence is due to the interplay between the non-convexity of the objective function and the fact that the set of functions expressed by threshold functions is larger (a consequence of lemma 2).

6. Conclusions

In this paper, we have presented a novel analysis of the expressiveness of rectifier neural networks. Specifically, for binary classification we showed that even though the decision boundary of two-layer rectifier network can be represented using threshold unit network, the number of threshold units required is exponential. Further, while a corresponding general logarithmic reduction of threshold units is not possible, for specific networks, we characterized sufficient conditions for reducing a threshold network to a much smaller rectifier network. We also presented a relaxed condition where we can approximately recover a rectifier network that is hidden layer equivalent to an exponentially larger threshold network.

Our work presents a natural next step: can we use the equivalence of the expressiveness results given in this paper to help us study the sample complexity of rectifier networks? Another open question is the generalization of these results to deep networks. Finally, from our experiments we see that expressiveness is not enough to guarantee learnability. Studying the interplay of expressiveness, sample complexity and the convexity properties of the training objective function for rectifier networks represents an exciting direction of future research.

Acknowledgements

We are grateful to the reviewers for their invaluable comments and pointers to related work. Xingyuan was partly supported by the School of Computing PhD fellowship.

[2] We also evaluated the performance on the training set. The training errors for all six datasets for all three learning scenarios are very close to test error, suggesting that over-fitting is not an issue.

References

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Blum, Avrim L. and Rivest, Ronald L. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1): 117 – 127, 1992.

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

Daniely, Amit, Linial, Nati, and Shalev-Shwartz, Shai. From average case complexity to improper learning complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, 2014.

DasGupta, Bhaskar and Schnitger, Georg. The power of approximating: a comparison of activation functions. In *Advances in Neural Information Processing Systems 5*, pp. 615–622. 1993.

Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Deep Sparse Rectifier Neural Networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.

Hajnal, András, Maass, Wolfgang, Pudlák, Pavel, Szegedy, Mario, and Turán, György. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46(2):129–154, 1993.

Hochreiter, Sepp. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998.

Klivans, A. R. and Sherstov, A. A. Cryptographic hardness for learning intersections of halfspaces. In *47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. 2012.

Livni, Roi, Shalev-Shwartz, Shai, and Shamir, Ohad. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems 27*, pp. 855–863. 2014.

Maas, Andrew L., Hannun, Awni Y., and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

Maass, W., Schnitger, G., and Sontag, E. D. On the computational power of sigmoid versus boolean threshold circuits. In *32nd Annual Symposium on Foundations of Computer Science*, 1991.

Maass, Wolfgang, Schnitger, Georg, and Sontag, Eduardo D. A comparison of the computational power of sigmoid and boolean threshold circuits. In *Theoretical Advances in Neural Computation and Learning*, pp. 127–151. 1994.

Martens, James, Chattopadhya, Arkadev, Pitassi, Toni, and Zemel, Richard. On the representational efficiency of restricted boltzmann machines. In *Advances in Neural Information Processing Systems 26*, pp. 2877–2885. 2013.

Montufar, Guido F, Pascanu, Razvan, Cho, Kyunghyun, and Bengio, Yoshua. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems 27*, pp. 2924–2932. 2014.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

Pascanu, Razvan, Montufar, Guido, and Bengio, Yoshua. On the number of response regions of deep feedforward networks with piecewise linear activations. In *International Conference on Learning Representations*, 2014.

Telgarsky, Matus. Representation Benefits of Deep Feedforward Networks. *arXiv:1509.01801*, 2015.

Zaslavsky, Thomas. *Facing up to arrangements: face-count formulas for partitions of space by hyperplanes*. American Mathematical Society, 1975.

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., and Hinton, G. E. On rectified linear units for speech processing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.