Neurocomputing 000 (2018) 1–6

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Efficiency of deep networks for radially symmetric functions

Brendan McCane*, Lech Szymanski

Department of Computer Science, University of Otago, Dunedin, New Zealand

ARTICLE INFO

Article history:
Received 28 July 2017
Revised 8 May 2018
Accepted 11 June 2018
Available online xxx

Communicated by Dr Q Wei

Keywords:
Deep networks
Function approximation

ABSTRACT

We prove that radially symmetric functions in d dimensions can be approximated by a deep network with fewer neurons than the previously best known result. Our results are much more efficient in terms of the support radius of the radial function and the error of approximation. Our proofs are all constructive and we specify the network architecture and almost all of the weights. The method relies on space-folding transformations that allow us to approximate the norm of a high dimensional vector using relatively few neurons.

1. Introduction

Deep networks have been stunningly successful in many machine learning domains since the area was reinvigorated by the work of Krizhevsky et al. [6]. Despite their success, relatively little is known about them theoretically, although this is changing. In particular, it would be very useful to know theoretically for which problems deep networks are more effective than shallow learners. This paper extends previous work on approximating radially symmetric functions. It provides a new upper bound for the number of neurons required in a deep network with rectified linear units (ReLUs) to approximate a radially symmetric function and does so using a constructive proof. A method for building ReLU networks that do the approximation is given.

2. Related work

Most theoretical work on deep networks consists of existence proofs that give no insight into how to build a network for the problem under consideration. For example, ReLU networks with n_0 inputs, L hidden layers of width $n \geq n_0$ can compute functions that have $\Omega\left((n/n_0)^{(L-1)n_0} n^{n_0}\right)$ linear regions compared to $\sum_{j=0}^{n_0} \binom{n}{j}$ for a shallow network [7]. More generally, Telgarsky [10] proved for semi-algebraic neurons (including ReLU, sigmoid etc), that networks exist with $\Theta(k^3)$ layers and up to a constant number of nodes per layer that require at least 2^k nodes to approximate with a network of $O(k)$ layers. Delalleau and Bengio [3] show that deep sum-product networks exist for which a shallow network would require exponentially more neurons to simulate. For convolutional arithmetic circuits (similar to sum-product networks), Cohen et al. [2], in an important result, show that "besides a negligible (zero measure) set, all functions that can be realized by a deep network of polynomial size, require exponential size in order to be realized, or even approximated, by a shallow network."

The above works, except for Cohen et al. [2] focus on approximating deep networks with shallow networks, but do not indicate what problems are best attacked with deep networks. For manifolds, Basri and Jacobs [1] show how deep networks can efficiently represent low-dimensional manifolds and that these networks are almost optimal, but they do not discuss limitations of shallow networks on the same problem. Somewhat similarly, Shaham et al. [8] show that depth-4 networks can approximate a function on a manifold where the number of neurons depends on the complexity of the function and the dimensionality of the manifold and only weakly on the embedding dimension. Again, they do not discuss the limitations of shallow networks for this problem. Importantly, both of these results are constructive and allow one to actually build the network.

Szymanski and McCane [9] show that deep networks can approximate periodic functions of period P over $\{0, 1\}^N$ with $O(\log_2 N - \log_2 P)$ parameters versus $O(P\log_2 N)$ for shallow. Eldan and Shamir [4] show that networks with two hidden layers exist such that the network can approximate a radially symmetric function with $O(d^{19/4})$ neurons, whereas a network with 1 hidden layer requires at least $O(e^d)$ neurons. They do not extend the result to deeper networks.

Therefore evidence is building that deep networks are more powerful than their shallow counterparts in terms of the number of parameters or neurons required. Nevertheless, more work is needed. In particular, it would be useful to determine which

* Corresponding author.
E-mail address: mccane@cs.otago.ac.nz (B. McCane).

https://doi.org/10.1016/j.neucom.2018.06.003
0925-2312/

problems are best solved with deep networks, and how to build networks for those particular problems. In this work we directly extend the work of Szymanski and McCane [9] and Eldan and Shamir [4]. The latter work [4] is extended to deeper networks for approximating radially symmetric functions that require fewer parameters than their shallow counterparts. The former [9] is extended by generalising their notion of folding transformations to work in multiple dimensions and more simply with ReLU networks. The proofs are constructive and allow us to build networks for approximating radially symmetric functions.

3. Context and notation

A radially symmetric function is a function whose value is dependent on the norm of the input only. We are interested in L-Lipschitz functions f, $|f(x) - f(y)| \leq L|x - y|$, as this covers many functions common in classification tasks. The number of dimensions of the input is d, and we assume that f is constant outside a radius R. This is a similar context to that used by Eldan and Shamir [4]. Further, we restrict ourselves to ReLU networks only, which is more restrictive than Eldan and Shamir [4], but allows us to explicitly construct the networks of interest. A network with N layers has $N-1$ hidden layers. Layer 0 is the input layer (not counted in the number of layers), and layer N is the output layer.

Proofs are only sketched in the main body of the paper. Detailed proofs are provided in the supplementary material.

4. 3 Layer networks

We start by stating a modified form of Lemma 18 from Eldan and Shamir [4] to do with 3 layer networks:

Lemma 1 (Modified form of Lemma 18 from Eldan and Shamir [4]). *Let $\sigma(z) = \max(0, z)$. Let $f : \mathbb{R} \to \mathbb{R}$ be an L-Lipschitz function supported on $[0, R]$. Then for any $\delta > 0$, there exists a function $g : \mathbb{R}^d \to \mathbb{R}$ expressible by a 3-layer network of width at most $\frac{6d^2R^2+3RL}{\delta}$, such that*

$$\sup_{\boldsymbol{x}\in\mathbb{R}^d} |g(\boldsymbol{x}) - f(||\boldsymbol{x}||)| < \delta + L\sqrt{\delta}$$

The proof follows the basic plan of Eldan and Shamir [4] where the first layer is the input layer, the second layer approximates x_i^2 for each dimension i, and the third layer computes $\sum_i x_i^2$ and approximates f. Since several sections of the second layer are doing the same thing (computing the square of their input), a weight-sharing corollary follows immediately where only one copy of the square approximation is needed.

Corollary 1 (3 Layer Weight Sharing). *Let $\sigma(z) = \max(0, z)$. Let $f : \mathbb{R} \to \mathbb{R}$ be an L-Lipschitz function supported on $[0, R]$. Then for any $\delta > 0$, there exists a function $g : \mathbb{R}^d \to \mathbb{R}$ expressible by a 3-layer weight-sharing network with at most $\frac{6\ dR^2+3RL}{\delta}$ weights, such that*

$$\sup_{\boldsymbol{x}\in\mathbb{R}^d} |g(\boldsymbol{x}) - f(||\boldsymbol{x}||)| < \delta + L\sqrt{\delta}$$

5. Deep folding networks

In this section, we show how folding transformations can be used to create a much deeper network with the same error, but many fewer weights than needed in Lemma 1. A folding transformation is one in which half of a space is reflected about a hyperplane, and the other half remains unchanged. Fig. 1 a demonstrates how a sequence of folding transformations can transform a circle in 2D to a small sector. After enough folds, we can discard the almost zero coordinates to approximate the norm. We will use this general idea to prove the following theorem:

Theorem 1. *Let $\boldsymbol{x} \in \mathbb{R}^d$, and $\sigma(z) = \max(0, z)$. Let $f : \mathbb{R} \to \mathbb{R}$ be an L-Lipschitz function supported on $[0, R]$. Fix L, δ, $R > 0$. There exists a function $g : \mathbb{R}^d \to \mathbb{R}$ expressible by a $O(d\log_2(d) + \log_2(d)\log_2(\frac{R}{\sqrt{\delta}}))$ layer network where the number of weights, and number of neurons, $N_w, N_n = O(d^2 + d\log_2(\frac{R}{\sqrt{\delta}}) + \frac{3RL}{\delta})$, such that:*

$$\sup_{\boldsymbol{x}\in\mathbb{R}^d} |g(\boldsymbol{x}) - f(||\boldsymbol{x}||)| < \delta + L\sqrt{\delta}$$

The approach taken here is a constructive one and specifies the architecture of the network needed to approximate f. In fact, all of the weights are specified by the construction. The approach is somewhat different to that used to prove Lemma 1. We build a sequence of layers to directly approximate $||\boldsymbol{x}||$ and then approximate f in the last layer. To build our layers, we need a few helper lemmas.

Lemma 2 (2D fold). *There exists a function $g : \mathbb{R}^2 \to \mathbb{R}^2$, expressible by a ReLU network with 4 ReLU units and 2 sum units that can compute a folding transformation about a line through the origin, represented by the unit direction vector $\boldsymbol{l} = (l_x, l_y)^T$. The function g is of the form:*

$$g(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} & \boldsymbol{l}\cdot\boldsymbol{x}^\perp > 0 \\ \begin{bmatrix} l_x^2 - l_y^2 & 2l_xl_y \\ 2l_xl_y & l_y^2 - l_x^2 \end{bmatrix}\boldsymbol{x} & otherwise \end{cases}$$

The requisite ReLU network is shown in Fig. 1b. Only one of the nodes labeled x_- (y_-) and x_+ (y_+) are active at any one time. Therefore there are four possible cases depending on which two nodes are active. Note that x_- is active when $\boldsymbol{l}\cdot\boldsymbol{x}^\perp < 0$ and x_+ is active when $\boldsymbol{l}\cdot\boldsymbol{x}^\perp > 0$. To approximate the 2D norm, we simply stack layers of the type shown in Fig. 1b with suitable choice of l_x, l_y at each layer. Note that the summation nodes are not required since they can be incorporated into the summations and weights of the next ReLU layer. These 2D folds can be used to estimate the norm of a vector as per the following lemma.

Lemma 3 (Approximate $||\mathbf{x}||, x \in \mathbb{R}^2, ||x|| < R$). *There exists a function $g : \mathbb{R}^2 \to \mathbb{R}$, expressible by a ReLU network with no more than $\log_2\left(R\frac{\pi}{\delta}\right)$ layers and 4 nodes per layer such that:*

$$\sup_{\boldsymbol{x}\in\mathbb{R}^2, ||\boldsymbol{x}||\leq R} |g(\boldsymbol{x}) - ||\boldsymbol{x}||| \leq \delta$$

Proof. The proof is short and simple. After f layers, each data point will be within an angle of $\frac{\pi}{2^f}$ of the x-axis. Simple geometry and appropriate approximations leads to:

$$\begin{aligned} \delta &= ||\boldsymbol{x}|| - ||\boldsymbol{x}||\cos\left(\frac{\pi}{2^f}\right) \\ &\leq R\left(1 - \cos\left(\frac{\pi}{2^f}\right)\right) \\ &\leq R\left(2\sin\left(\frac{\pi}{2^{f+1}}\right)\right) \\ &\leq R\left(\frac{\pi}{2^f}\right) \\ f &\leq \log_2\left(R\frac{\pi}{\delta}\right) \end{aligned}$$

□

The following lemma generalises this construction to folds in d dimensions.

Lemma 4 (Approximate $||\mathbf{x}||, x \in \mathbb{R}^d$). *There exists a function $g : \mathbb{R}^d \to \mathbb{R}$, expressible by a ReLU network with:*

$$N_l \leq \log_2(d)\log_2\left(\frac{R\pi}{\delta}\left[(2^{\lfloor\frac{(d+1)}{2}\rfloor} - 1) + \sqrt{2}(2^{\lfloor\frac{d}{2}\rfloor} - 1)\right]\right)$$

Please cite this article as: B. McCane, L. Szymanski, Efficiency of deep networks for radially symmetric functions, Neurocomputing (2018), https://doi.org/10.1016/j.neucom.2018.06.003