A closer look at the impact of activation functions on Generalizability

Gabriela Tsvetkova
gabriellatsvetkova43@gmail.com

Under the guidance of Victor Kolev
Stanford University

2020

Content

1 Introduction

Over the past decades, an overwhelming body of research has been constituting the effectiveness of Deep Learning. These black-box models - neural networks - are not explicitly programmed for a particular task, instead an ability to optimize their performance is implemented. They do so as they gain experience starting from zero, and gradually acquire potential for recognizing certain patterns.

Neural networks are basically a complex composition of affine transformations. However, linearity does not give the model the capacity needed to approximate any function given unlimited data and infinite time for training. That is why non-linear activation functions such as Sigmoid, Tanh, ReLU are introduced to the problem.

A recent paper [4] showed that using $y = sin(x)$ as an activation function keeps the structure of the training examples because $\frac{d^n}{d\ x^n} \sin(\theta) = \cos(\theta), n \in \mathbb{N}$.

In this paper we investigate the influence of 3 activation functions while training the models to approximate fundamental mathematical functions such as $y = x^2$ and $y = x^3$. Empirically, it can be inferred that activation functions implement explicit structures in the network, which biases training towards functions similar to the activation first derivatives.

This work paves way for research into the inductive biases of neural network design, which if properly exploited could improve network stability, training speed, generalization, and applicability.

2 Concepts

2.1 Feedforward neural network

A feedforward neural network consists of nodes and connections between them. Each neuron is associated with an activation and a bias and each connection – with a certain weight.

The value of the activation function of the neurons in layer l is obtained by the following formula:

$$\boldsymbol{a}^{(l)} = g(\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)})$$

Here, n_l is the number of units in layer l; $\boldsymbol{a}^{(l)} \in \mathbb{R}^{n_l}$ is the vector containing the activations of the neurons in layer l; $\boldsymbol{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ is the matrix containing the weights which connect the neurons in layer l to those in layer $l-1$; $\boldsymbol{b}^{(l)} \in \mathbb{R}^{n_l}$ is the vector containing the biases of the neurons in layer l; and g is some non-linear activation function.

2.2 Loss function

The loss function evaluates how well specific algorithm models the given data i.e. the error. Since we are working in a continuous domain, we use the mean

square error function:

$$\mathcal{L} = \frac{1}{n_L}\|\boldsymbol{y} - \boldsymbol{a}^{(L)}\|^2$$

Here, $\boldsymbol{y} \in \mathbb{R}^{n_L}$ is the vector containing the ground truth outputs; $\boldsymbol{a}^{(L)} \in \mathbb{R}^{n_L}$ is the vector containing the activations of the neurons in the last layer (L) i.e. the vector containing the predictions; and $\|.\|$ is the Euclidean norm. Two algorithms are involved in the process of learning of the neural network – backpropagation and gradient descent. For the latter we use Adam [2] in our experiments.

3 Methods

3.1 Activation Functions

In this paper the activation functions used are as follows: ReLU, Tanh, Leaky ReLU, and SiNU:

$$\text{ReLU}(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

$$\text{Leaky ReLU}(x) = \begin{cases} 0.01x & x \leq 0 \\ x & x > 0 \end{cases}$$

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\text{SiNU}(x) = sin(x)$$

3.2 Spectral Norm Regularization

Neural networks tends to learn the training examples well, but fail to generalize to new examples. This is known as overfitting, and stems from the fact that the minimum error objective is ill-posed. For instance, the Lagrange polynomial of a set of points would have the smallest loss, yet it would not have high generalization error.

One method of alleviating overfitting is regularization. This is the process of penalizing big values in the weight matrix with the effect of a model which adapts well to new data.

In this paper we focus on Spectral Norm Regularization. [6]. The motivation behind it is due to (Neyshaubar etal 2019)[3] and [1]. The spectral norm of a matrix $\text{W} \in \mathbb{R}^{m \times n}$ is the largest singular value of it and it is defined as:

$$\sigma = \max_{\xi \in \mathbb{R}^n, \xi \neq 0} \frac{\|A\xi\|_2}{\|\xi\|_2}$$

4 Results

We use fully-connected layers as size, range, activation function, spectral norm regularization coefficient. In the graphs below are shown the models with the best performance on the two tasks.

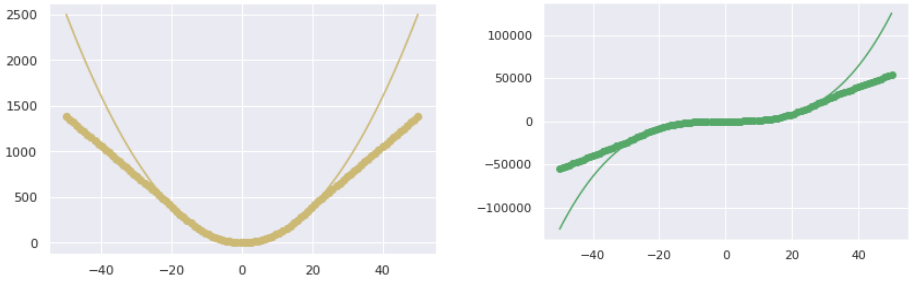

Figure 1: The left graph shows a model that approximates $y = x^2$ trained on range [-20;20] with ReLU. It is visible that the graph of the objective function keeps a linear structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

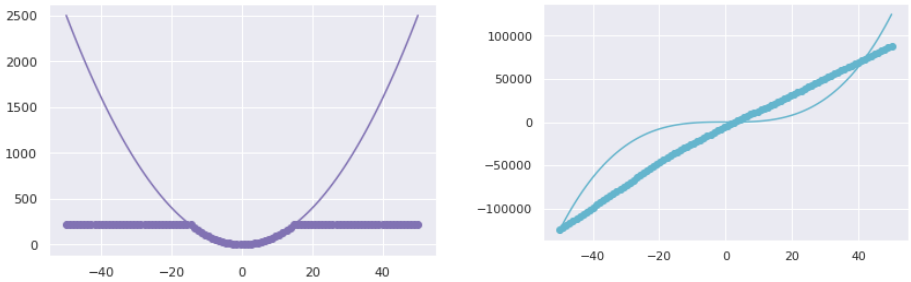

Figure 2: The left graph shows a model that approximates $y = x^2$ trained on range [-20;20] with Tanh. It is visible that the graph of the objective function keeps a linear structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

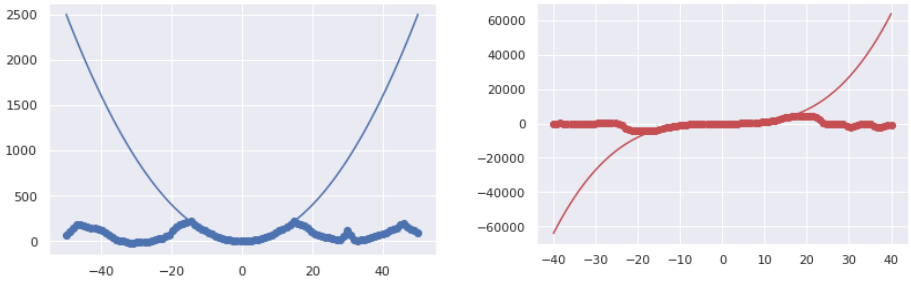

Figure 3: The left graph shows a model that approximates $y = x^2$ trained on range [-20;20] with SiNU. It is visible that the graph of the objective function keeps a linear structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

5 Future Development

Experiments such as changing the range of input data, type of function, the architecture of the neural network and the type of regularization Dropout[5] are always interesting and may lead to better understanding of the impact of activation functions. A further investigation with modular neural networks [7] for composing complex functions can be made.

6 Conclusion

References

[1] Sanjeev Arora et al. "Stronger generalization bounds for deep nets via a compression approach". In: *CoRR* abs/1802.05296 (2018). arXiv: 1802.05296. URL: http://arxiv.org/abs/1802.05296.

[2] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[3] Behnam Neyshabur et al. "Exploring Generalization in Deep Learning". In: *CoRR* abs/1706.08947 (2017). arXiv: 1706.08947. URL: http://arxiv.org/abs/1706.08947.

[4] Vincent Sitzmann et al. *Implicit Neural Representations with Periodic Activation Functions*. 2020. arXiv: 2006.09661 [cs.CV].

[5] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[6] Yuichi Yoshida and Takeru Miyato. *Spectral Norm Regularization for Improving the Generalizability of Deep Learning*. 2017. arXiv: 1705.10941 [stat.ML].

[7] Jie Zhou et al. "Graph Neural Networks: A Review of Methods and Applications". In: *CoRR* abs/1812.08434 (2018). arXiv: 1812.08434. URL: http://arxiv.org/abs/1812.08434.