CMSE 890-002: Mathematics of Deep Learning, MSU, Spring 2020

Lecture 14: Vanishing Gradients and Approximation Theory

February 12, 2020

Lecturer: Matthew Hirn

7.3 Why ReLU? Vanishing gradients and optimization

Classically, sigmoidal type nonlinear activation functions have been used in neural networks because they mimic the perceptron, which is thought to to be a good biological model for neural activation in the brain. On the other hand, recently the ReLU activation function $\sigma(z) = \max(0, z)$ and variants of it have become very popular. One explanation for this is in the optimization of neural networks, and the phenomenon of "vanishing gradients."

Consider a simple neural network in which $x \in \mathbb{R}$, all weights and biases are also scalar valued, and we have $L = 2$,

$$f(x;\theta) = \alpha\sigma_2(w_2\sigma_1(w_1x + b_1) + b_2)\,.$$

Let us compute the partial derivative of $f(x;\theta)$ with respect to w_1; we obtain:

$$\partial_{w_1} f(x;\theta) = \alpha\sigma_2'(w_2\sigma_1(w_1x + b_1) + b_2)w_2\sigma_1'(w_1x + b_1)x\,.$$

More generally, for L layers but with the same scalar valued weights and biases, we would have:

$$\partial_{w_1} f(x;\theta) = \alpha \cdot \left(\prod_{\ell=2}^{L} w_\ell\right) \cdot \left(\prod_{\ell=1}^{L} \sigma_\ell'(z_\ell)\right) \cdot x\,, \tag{23}$$

for values

$$z_\ell = A_\ell(\sigma_{\ell-1}(A_{\ell-1}(\cdots\sigma_1(A_1(x))))\,.$$

Notice that if σ_ℓ is a sigmoid or hyperbolic tangent, that $\sigma_\ell'(z_\ell)$ will be small if $|z_\ell| \gg 0$. Furthermore, if several such values $\sigma_\ell'(z_\ell)$ for multiple ℓ are small, we will multiply them together, making them even smaller. Thus the partial derivative in the w_1 weight will be nearly zero, making it very hard for the gradient descent algorithm to change this weight. Similar analysis holds for the other weights, although less multiplications will be involved. This phenomenon is referred to as the *vanishing gradient* problem in deep learning, and affects all neural networks, not just scalar valued ones.

The ReLU function helps to alleviate the the vanishing gradient issue and thus make training easier since $\sigma'(z) = 1$ for $z > 0$ if $\sigma(z) = \max(0, z)$. Thus the gradient will not vanish due to σ. Equation (23) indicates one also needs to be careful with the weights, making sure they do not get too small either, since the values of all the weights affects the gradient of w_1. On the other hand, if the weights get too large, their multiplication will be

a very large number, and the gradient will explode, leading to a very unstable optimization. Proper initialization of the training data and weights help with these matters, as do other heuristics in deep learning, that we will not go into but may be of interest to the reader.

We will see in later sections, that ReLU is also a good choice from the perspective of approximation theory.

8 Classic approximation theory results

We now shift into approximation theoretic results for artificial neural networks, which was an active area of research in the late 1980s and 1990s, and with the recent popularity of neural networks has re-emerged as an important topic in mathematics and computer science. The field of approximation theory is concerned with one's ability to approximate complex functions with a collection of simpler functions.

In our case, we will focus on the data space $\mathcal{X} = [0,1]^d$, the complex functions will be a deterministic label function

$$y = F(x)\,, \quad x \in [0,1]^d\,,$$

and the simpler functions will be one layer neural networks

$$\mathcal{F} = \{f(x;\theta) = f(x;\mathbf{W},b,\alpha)\}\,,$$

of the form (22). Initially we will discuss results of the following form: given $F(x)$ and $\epsilon > 0$, does there exists a neural network $f(x;\theta)$ (meaning does there exist a number of layers $L < \infty$ and a finite number of parameters θ) such that

$$|F(x) - f(x;\theta)| < \epsilon\,, \quad \text{for all } x \in \mathcal{X}\,?$$

Recalling our squared loss we will also consider the following: for each $\epsilon > 0$ and $F(x)$ does there exist a neural network $f(x;\theta)$ such that

$$\mathbb{E}_X[(F(X) - f(X;\theta))^2] = \int_{\mathcal{X}} (F(x) - f(x;\theta))^2 p_X(x)\,dx < \epsilon\,?$$

We will initially just be concerned with whether the answer to these questions is true or not. We will see that for certain classes of functions and certain neural network architectures the answer is indeed in the affirmative. Then we will investigate further and try to interpret and relate such results to "real life," which often will motivate the search for better theory.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*, 2015.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

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

[4] Larry Greenemeier. AI versus AI: Self-taught AlphaGo Zero vanquishes its predecessor. *Scientific American*, October 18, 2017.

[5] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. arXiv:1803.08823, 2018.

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer-Verlag New York, 2nd edition, 2009.

[7] J. Hartlap, P. Simon, and P. Schneider. Why your model parameter confidences might be too optimistic - unbiased estimation of the inverse covariance matrix. *Astronomy and Astrophysics*, 464(1):399–404, 2007.

[8] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, 2002.

[9] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[10] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, San Diego, CA, USA, 2015.

[11] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[12] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.