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

Lecture 17: One-Layer Networks Continued

February 19, 2020

Lecturer: Matthew Hirn

From the perspective of classical approximation theory, one way to resolve this issue is to assume that $F(x)$ is smoother, say $F \in \mathbf{C}^{s+1}[0,1]^d$, which means that F and all derivatives of F up to order $(s+1)$ are continuous and bounded. In this case one can approximate $F(x)$ in a neighborhood of $u \in \mathbb{R}^d$ by a polynomial $p_u(x)$, which is the Taylor polynomial of F around u, i.e.,

$$p_u(x) = \sum_{\|\beta\|_1 \leq s} \frac{1}{\beta!} (\partial^\beta F)(u)(x-u)^\beta \, ,$$

where as in Example 8.4 $\beta = (\beta(1), \ldots, \beta(d)) \in \mathbb{N}^d$ with $\|\beta\|_1$ and x^β defined the same way. We also define

$$\beta! = \prod_{k=1}^{d} \beta(k)! \, ,$$

and

$$(\partial^\beta F)(x) = [(\partial_1)^{\beta(1)} \cdots (\partial_d)^{\beta(d)} F](x) \, .$$

The approximation property of $p_u(x)$ can be quantified using Taylor's theorem, which gives

$$|F(x) - p_u(x)| \leq C \|x-u\|^s \, , \tag{29}$$

where C is the $\mathbf{C}^{s+1}[0,1]^d$ norm of F; that is,

$$C = \max_{\|\beta\|_1 \leq s+1} \sup_{x \in [0,1]^d} |\partial^\beta F(x)| \, .$$

From (29) it follows that

$$\|x-u\| \leq \epsilon^{1/s} \quad \Longrightarrow \quad |F(x) - p_u(x)| \leq C\epsilon \, .$$

Suppose then that instead of a one-layer ReLU network, which implements a local linear regression, we instead developed a one layer neural network in which each neuron implements a local polynomial regression. The previous equation shows that in order to approximate $F(x)$ uniformly on $[0,1]^d$ with a $C\epsilon$ error, we would need to have a training set that samples $[0,1]^d$ on an $\epsilon^{1/s}$ grid, which means that

$$\#\text{ of training points} \; = \; \#\text{ of neurons} + 1 = O(\epsilon^{-d/s}) \, .$$

This is definitely an improvement over the one layer ReLU network, but is still inadequate. Indeed, the dimension d is often very large and the smoothness s is rarely proportional to d, meaning that $s \ll d$ and thus that the number of training points / the number of neurons is still astronomical. For future reference, we remark that if the number of training points $N+1$ is fixed, and hence the number of "local polynomial neurons" is N, we obtain that there exists such a one layer neural network $f_N(x;\theta)$ with

$$\sup_{x\in[0,1]^d} |F(x) - f_N(x;\theta)| < CN^{-s/d}\,. \tag{30}$$

Thus as the number of training points $N \to \infty$, the approximation improves and the rate at which the error goes to zero speeds up with increased smoothness (i.e., the larger s, the faster the convergence) and decreases with dimension (i.e., the larger d, the slower the convergence).

8.3 Refined one-layer analysis and two-layer neural networks

Our goal in this section is to arrive at a seemingly remarkable two-layer result of Pinkus that can be found in [13].

8.3.1 Refinements of the one layer analysis

We first extend the one layer results to more general activation functions, and also refine their analysis. Recall the approximation theorems of Cybenko (Theorem 8.1 and 8.2) and Hornik (Theorem 8.3) relied on σ being either sigmoidal or bounded and non-constant, while the analysis in the previous section used the ReLU activation specifically. In fact, it turns out that any continuous σ that is not a polynomial works. Let us explain.

In order to simplify the exposition, note that any one layer neural network of the form

$$f(x;\theta) = f(x;\mathbf{W},b,\alpha) = \sum_{k=1}^{m} \alpha(k)\sigma(\langle x, w_k\rangle + b(k))\,, \tag{31}$$

lies in the space

$$\mathcal{M}(\sigma) = \operatorname{span}\{\sigma(\langle x, w\rangle + b) : w \in \mathbb{R}^d\,,\, b\in\mathbb{R}\}\,,$$

and vice versa, that is any function in $M(\sigma)$ is a one-layer neural network of the form (31). Theorem 8.1 and variants of it are then concerned with the following question: For which $\sigma : \mathbb{R} \to \mathbb{R}$ is it true that, for any $F \in \mathbf{C}[0,1]^d$ and $\epsilon > 0$, there exists a function $f(x;\theta) \in M(\sigma)$ such that

$$\sup_{x\in[0,1]^d} |F(x) - f(x;\theta)| < \epsilon\,?$$

Theorem 8.1 proves the result for $\sigma(z)$ that are continuous and sigmoidal. It turns out that the result is try for much more general functions σ.

Theorem 8.7 (Pinkus 1999, [13]). *Let $\sigma(z)$ be a continuous function. If $\sigma(z)$ is not polynomial, then for each $F \in \mathbf{C}[0,1]^d$ and each $\epsilon > 0$, there exists an $f \in M(\sigma)$ such that*

$$\sup_{x\in[0,1]^d} |F(x) - f(x;\theta)| < \epsilon\,. \tag{32}$$

Conversely, if for each $F \in \mathbf{C}[0,1]^d$ and each $\epsilon > 0$ there exists an $f \in M(\sigma)$ such that (32) holds, then $\sigma(z)$ is not a polynomial.

Theorem 8.7 strengthens Theorem 8.1 by allowing for any continuous activation function $\sigma(z)$ so long as it is not a polynomial. The converse is relatively trivial. Indeed, if $\sigma(z)$ were a polynomial of degree m, then $M(\sigma)$ would be the space of all degree m or less polynomials, which certainly cannot approximate any continuous function (simply take F as a polynomial of degree strictly larger than m). Let us sketch the proof of Theorem 8.7, as it contains some interesting points.

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.

[13] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[14] Vitaly E. Maiorov. On best approximation by ridge functions. *Journal of Approximation Theory*, 99:68–94, 1999.

[15] Vitaly E. Maiorov and Allan Pinkus. Lower bounds for approximation by mlp neural networks. *Neurocomputing*, 25:81–91, 1999.

[16] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 907–940. PMLR, 2016.