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

Lecture 12: Artificial Neural Networks

February 7, 2020

Lecturer: Matthew Hirn

A mathematical neuron takes in a vector $x \in \mathbb{R}^d$ and processes it via an affine function into $\mathbb{R}$ followed by a nonlinear "activation" function, that responds based on the output of the affine function. A single layer of a neural network has many such neurons. For example, suppose we have d_1 such neurons,

$$\eta_k(z) = \sigma(\langle x, w_k \rangle + b(k)) \,, \quad w_k \in \mathbb{R}^d \,, \, b(k) \in \mathbb{R} \,, \, 1 \leq k \leq d_1 \,.$$

We collect the affine parts of these d_1 neurons into a single affine map $A : \mathbb{R}^d \to \mathbb{R}^{d_1}$,

$$A(x) = (\langle x, w_1 \rangle + b(1), \ldots, \langle x, w_{d_1} \rangle + b(d_1))^T \in \mathbb{R}^{d_1 \times 1} \,.$$

If we collect the weights w_k into a single $d \times d_1$ matrix $\mathbf{W}$ and the biases $b(k)$ into a single $d_1 \times 1$ vector b,

$$\mathbf{W} = \begin{pmatrix} | & & | \\ w_1 & \cdots & w_{d_1} \\ | & & | \end{pmatrix} \qquad b = \begin{pmatrix} b(1) \\ \vdots \\ b(d_1) \end{pmatrix} \,,$$

then we can rewrite $A(x)$ as:

$$A(x) = \mathbf{W}^T x + b \,.$$

We can also write, with a slight abuse of notation, the collection of all d_1 neurons as the map:

$$x \mapsto \sigma(A(x)) = \sigma(\mathbf{W}^T x + b) = (\eta_1(x), \ldots, \eta_{d_1}(x))^T \,,$$

where it is understood that σ acts element-wise.

An *artificial neural network* cascades the operations $\sigma(A(x))$ multiple times. Suppose we have L layers and the affine transform at each layer is denoted by A_ℓ, $1 \leq \ell \leq L$, in which $A_1 : \mathbb{R}^d \to \mathbb{R}^{d_1}$ and $A_\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$ for $2 \leq \ell \leq L$. An artificial neural network is a representation $\Phi(x; \boldsymbol{\theta}) = \Phi_{\text{ANN}}(x; \boldsymbol{\theta})$ given by:

$$\Phi(x; \boldsymbol{\theta}) = \sigma_L(A_L(\cdots \sigma_1(A_1(x)))) \,,$$

where the representation has parameters $\boldsymbol{\theta}$ that are encoding the weight matrices $\mathbf{W}_1, \ldots, \mathbf{W}_L$ and the bias vectors $b_1, \ldots, b_L$. There is often a final layer that takes the vector $\Phi(x; \boldsymbol{\theta}) \in \mathbb{R}^{d_L}$ and maps it to a scalar for regression or classification. In the regression setting, this means we have a final weight vector $\alpha \in \mathbb{R}^{d_L \times 1}$ such that

$$f(x; \boldsymbol{\theta}, \alpha) = \langle \Phi(x; \boldsymbol{\theta}), \alpha \rangle \,. \tag{22}$$

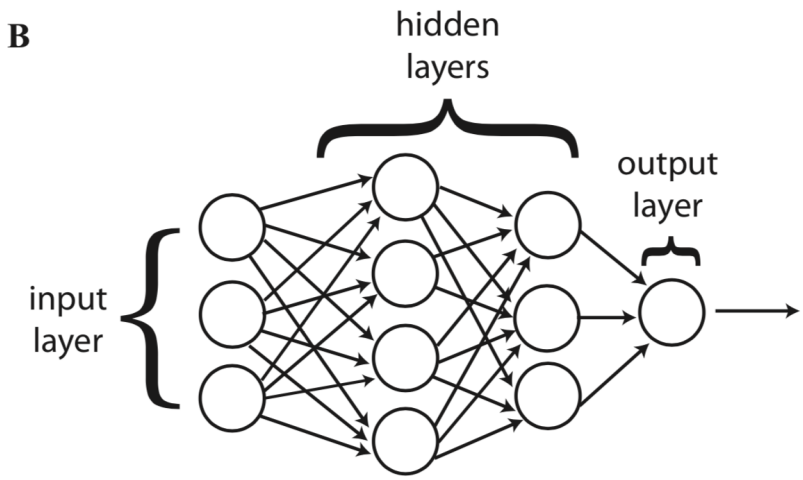

Figure 23: *An artificial neural network in with an input data point (layer)* $x \in \mathbb{R}^3$, $A_1 : \mathbb{R}^3 \to \mathbb{R}^4$, $A_2 : \mathbb{R}^4 \to \mathbb{R}^3$, *and* $\theta_1 \in \mathbb{R}^3$. *Figure taken from [5].*

Similarly, for the logistic regression (can also be generalized to softmax regression), we have:

$$p(y \mid x, \boldsymbol{\theta}, \alpha) = \frac{1}{1 + e^{-\langle \Phi(x;\boldsymbol{\theta}), \alpha \rangle}} .$$

The input to the network, $x \in \mathbb{R}^d$, is often called the *input layer*, the output $f(x; \boldsymbol{\theta}, \alpha)$ or $p(y \mid x, \boldsymbol{\theta}, \alpha)$ the *output layer*, and the layers in between the *hidden layers*. Figure 23 depicts an artificial neural network, and here is a two layer network for regression expanded out:

$$f(x; \boldsymbol{\theta}, \alpha) = \alpha^T \sigma_2(\mathbf{W}_2^T \sigma_1(\mathbf{W}_1^T x + b_1) + b_2) .$$

The compositional structure of neural networks is thought to be one of the keys to their success. Indeed, unlike linear models that have no composition and logistic regression which only has one affine transform and one nonlinear sigmoid, neural networks can have L such layers of different widths and with different nonlinear functions at each layer. They thus have significantly more flexibility, particularly for larger L. We will explore this in more detail in later sections.

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.