64. Neural Networks

Abstract

One of the most common concepts surrounding artificial intelligence is the term "neural network." As this entry will discuss, the neural network is not only surprisingly simple, but it also illuminates the functionality and limitations of even the most advanced machine learning models used today. Stated simply, a neural network is a collection of mathematical functions working in unison to process numerical input in a way that produces correct, "right" output according to some user-defined goal (*e.g.*, ground truth).

Definition: A neural network is a collection of simple, singular functions that take numerical input, process that input through a series of mathematical operations (*e.g.*, multiplication or addition), and produce an output.

Introduction

Traditional computer programming requires an author to tell a machine exactly what to do. If a programmer wanted a machine to produce a sequence of numbers representing the Fibonacci Sequence (*e.g.*, 1, 2, 3, 5, 8, ...) then the programmer would have to tell the machine to loop on the current number and add to that number the prior number (*e.g.*, $2 + 1 = 3$, $3 + 2 = 5$, and so on). This requires specialized knowledge of how the Fibonacci Sequence works, and any computer "algorithm" created to produce this sequence without that knowledge will fail. Machine learning flips this paradigm on its head, requiring the programmer to know only the correct output, not how that output was generated. This magic is made possible, in part, by representing problems as mathematical equations—*i.e.*, networks that process input to arrive at correct output.

Neural networks are one of two fundamental features in machine learning that permit automated learning—digitizing the ability to solve future, unknown problems given examples of currently known solutions. The neural network represents the "machine" of machine learning (the second piece relates to "learning" which is equally important when making these networks useful). Although neural networks represent, as a singular artificial neuron, intuitive mathematics, the layering and scale found in real-world neural networks, not to mention the relation to neuroscience, may make the concept unfamiliar at first blush. This entry will provide a simple definition of neural networks, the historical context of neural networks paired with a toy example of a single-layer neural network, limitations of neural networks, and a few closing considerations.

Background

In the 1940s, Warren McCulloch, a neurophysiologist, and Walter Pitts, a logician, realized that one of the most advanced machines in existence, the human brain, relied on incredibly simple neurons that themselves could be represented as individual functions. A neuron receives input through a dendrite, processes that input in the nucleus, and produces an output sent along an axon; artificial neurons could be built to take input in the form of numbers (dendrite), process the input via some type of algorithm (nucleus), and pass the result of the neuron forward (axon). Although this sounds complicated, it relies on simple mathematics.

The following mathematical formula may be considered a single neuron, a single neural network, just before the activation function is applied.

$$\text{bias} + (\text{input}_x * \text{weight}_1) + (\text{input}_y * \text{weight}_2) = \text{intermediate output} \tag{1}$$

To spell it out, the intermediate output is created by taking some bias term (*e.g.*, an integer, like negative two that remains constant), two inputs (*i.e.*, an x value and a y value, values to be provided), and two sets of weights (*e.g.*, two integers, such as one for input_x and one for input_y, that also remain constant). This set of information is processed (*i.e.*, multiplication plus addition in this case) and will produce a number. That number, the intermediate output, is then run through an activation function. The activation function, like a ReLU for example, is also not complicated: If the intermediate output is less than zero then the final output is zero; if the intermediate output is more than or equal to zero, then the final output is one. In short, if each of these variables (*i.e.*, bias, input_x, input_y, weight_1, weight_2) is set just right, then the artificial neuron will produce correct output to a specific—and potentially useful—problem, like the representation of a logical "AND" gate (*i.e.*, the mathematical function above will produce correct output for the logical "AND" gate).1

Interestingly, although the neuron just described is the brain behind some of the most complex artificial intelligence architectures today, the idea went dormant in the decades after its release. This dormancy remained despite Frank Rosenblatt's work on automatically updating the neuron's variables to receive more and more correct output, representing a step forward that unlocked the potential of machine learning. As a quote from 1958 portrays: "We are now about to witness the birth of such a machine—a machine capable of perceiving, recognizing and identifying its surroundings without any human training or control." What Professor Rosenblatt was referring to, and what was not practically possible, given limited hardware support, until the recent artificial intelligence craze, was the universal approximation theorem: "[A]ny real-world problem which is able to be mathematically mapped as a continuous function can be solved with nearly-perfect accuracy by using a neural network." In other words, if a neural network is large enough, it can solve any real-world problem that may be mathematically described. To see why this works, consider the neuron stated above.

That simple neuron, a perceptron, if the correct values are supplied, will produce the correct output for a very specific problem, the logical "AND" gate. The "AND" gate should produce a <1> if given input of (1,1). If we set the bias term to negative two and both weights (*i.e.*, weight_1 and weight_2) to one, then our network will do just that.

$$\begin{aligned} -2 + (1 * 1) + (1 * 1) &= 0 \\ 0 &\geq 0 \\ \text{output:} &= 1 \end{aligned} \tag{2}$$

This output, zero, is more than or equal to zero, which means that the function produces a <1>. This is correct and works for all types of input the gate may receive, from (0,0) to (1,0) to (0,1).

But what if we did not want the network to represent the logical "AND" gate. What if instead we wanted the neural network to represent the logical "XOR" gate. A "XOR" gate should produce a <0> given an input of (1,1). Our current neuron does not work.

What Rosenblatt was considering when making the statement about the universal approximation theorem was the layering of one neuron with other neurons, something that can, theoretically, answer complex problems like the "XOR" gate. In this case, layering a similar type of mathematical function, a logical "NOT AND" neuron, with another simple mathematical function, a logical "OR" neuron, will represent the "XOR" gate. These layers are where the concept of a "deep" neural network comes from. The more layers, the deeper the network.

It may not seem like it, but because of the universal approximation theorem, adding more and more layers to a neural network allows a machine to drive vehicles, recognize speech, and talk—from lethal autonomous weapons systems (Warfare: International Law and Artificial Intelligence) to synthetic data (Synthetic Data: Privacy Considerations) to the autonomous vehicles we have on the roads today, all of this is made possible thanks to the neural network.

Limitations

It can be useful to think of the inherent limitations of neural networks when trying to understand the concept. Three are particularly important.

First, neural networks operate by considering input and producing output. If a problem cannot be molded into that workflow (*i.e.*, a mathematical input-to-output) then a neural network will be no good at solving the problem. Consider image generators. Although there are an infinite number of colors that exist in the real world, computers can only represent colors in a finite and limited way, using a series of red, green, and blue pixel values. Although color itself is continuous in the natural world, computers must downgrade those colors in the digital world to operate on them, something which, in some ways, is unavoidably reductionist.

Second, it can sometimes be useful to think of machine learning models as inscrutable, given the difficulty of tracing particular inputs to particular outputs. However, that thinking elides the fact that these networks are surely accessible in some fashion. For instance, while it is generally true that some degree of randomness plays a role in machine learning model output, neural networks do not produce truly random output. In simple terms, the neural network of the AND gate from above will not suddenly begin producing words instead of numbers; a neural network taught to recognize hand-written digits will not suddenly be able to distinguish between images of cats and dogs. Even "random" and untoward outcomes of advanced neural networks that have learned undesired behaviors, like Microsoft's Tay, have a traceable path.

Finally, neural networks have finite and mathematical goals in mind. The AND gate from above was told exactly what type of output was correct: <0> or <1>. The world, according to that neural network, consisted of no more than zeros and ones. The same is true of neural networks used to drive vehicles or generate art. The network is approximating a problem (*i.e.*, the universal approximation theorem) by rudimentarily turning that problem into a mathematical equation. What this means is that unless the mathematical problem incorporates values we as a society care about, like equality, then the network will simply not consider them. Developers must think carefully about how neural networks are used and what is—or is not—considered as input and output by a neural network.

Conclusion

It was not until recently that researchers began layering neurons in such a massive fashion that the aims of McCulloch, Pitts, and Rosenblatt came to fruition. These networks are beginning to match or outperform humans at a variety of tasks, though those breakthroughs are coming from scale rather than invention. True enough, network types (*i.e.*, how the network processes data, such as the "AND" neural network above being called "feedforward" given that input is only ever passed forward through the network) have been showing improvements in recent years. For instance, convolutional neural networks became popular for vision, and generative adversarial networks started a generative trend that has been recently replaced (and will again be replaced in the future) by diffusion models. And yet, the core of these networks, so far, has remained the groundwork laid by McCulloch, Pitts, and Rosenblatt.

This axis of innovation may change in the future. Systems that rely on "more compute" to achieve better performance (*i.e.*, the systems relying on the same fundamental neurons conceptualized in the 50s) may get an update in the future, and that may unlock performance boosts that seem limited when or if scale plateaus. For the moment, however, neural networks can still be abstracted, in some sense, to the perceptron from the 50s, meaning that the understanding laid out above will continue to capture the essence of artificial

intelligence for the foreseeable future.

References

[1] Nathan Reitinger and Amol Deshpande. Epsilon-differential privacy, and a two-step test for quantifying reidentification risk. *Jurimetrics*, 63:263–317, 2023.

[2] Nathan Reitinger. Artificial intelligence is like a perpetual stew. *American University Law Review*, 73(5):1535, 2024.

[3] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52(1–2):99–115, 1990. Reprint of original 1943 article in Bulletin of Mathematical Biophysics 5:115–133.

[4] Abien Fred Agarap. Deep learning using rectified linear units (ReLU). arXiv preprint, 2018. arXiv:1803.08375, http://arxiv.org/abs/1803.08375, accessed 13 June 2024.

[5] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[6] Frank Rosenblatt. The design of an intelligent automaton. *Research Trends*, 6(2), 1958.

[7] Jeremy Howard and Sylvain Gugger. *Deep Learning for Coders with fastai and PyTorch*. O'Reilly Media, Inc., 2020.

[8] Steven M. Bellovin, Preetam K. Dutta, and Nathan Reitinger. Privacy and synthetic datasets. *Stanford Technology Law Review*, 22:1, 2019.

[9] Marty J. Wolf, Keith W. Miller, and Frances S. Grodzinsky. Why we should have seen that coming: Comments on Microsoft's Tay "Experiment," and wider implications. *ACM SIGCAS Computers and Society*, 47(3):54–64, 2017. doi: 10.1145/3144592.3144598.

[10] Leopold Aschenbrenner. Situational awareness: The decade ahead. https://situational-awareness.ai, 2024. accessed 13 June 2024.

[11] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun. CNN learning. In *A Guide to Convolutional Neural Networks for Computer Vision*, Synthesis Lectures on Computer Vision. Morgan & Claypool Publishers, 2018.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680, 2014.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.