John Dukewich
Dr. Jayme Peacock
18 March 2020
Audience: Upper level undergraduate computer science students with backgrounds in calculus and linear algebra

Neural Networks

Definition

A neural network is a machine learning model that can approximate arbitrary mathematical functions using a series of vector operations. The output of a neural network is strictly determined by weight parameters that are internal to the model. Finding the weights that best approximate the function is the essential task when using neural networks. By approximating functions, neural networks are often used for two cases: regression and classification. In regression, the neural network will output some real number based upon the input. An example of regression could be a neural network that will output a predicted credit score based upon an input consisting of numerical features like income, age, loan debt, and so on. An example of classification would be categorizing a picture as a cat or a dog based on an input image (the values of the pixels in the image). Neural networks have found uses in many applications ranging from facial recognition to language translation.

Neural Networks as a Black Box

At the most abstract level, neural networks can be viewed as a black box function. Some form of a vector is passed into the neural network and another vector is received as an output. For example, a 784 by 1 dimensional vector may be used, representing the values of each pixel in a 28-by-28 pixel image of a handwritten digit. Then, the neural network performs various vector operations to eventually output a 10 by 1 vector, with each value representing the probability that the original 28-by-28 pixel image is the digit corresponding to the index of that value. See Figure 1 to see what neural networks accomplish on an abstract level.

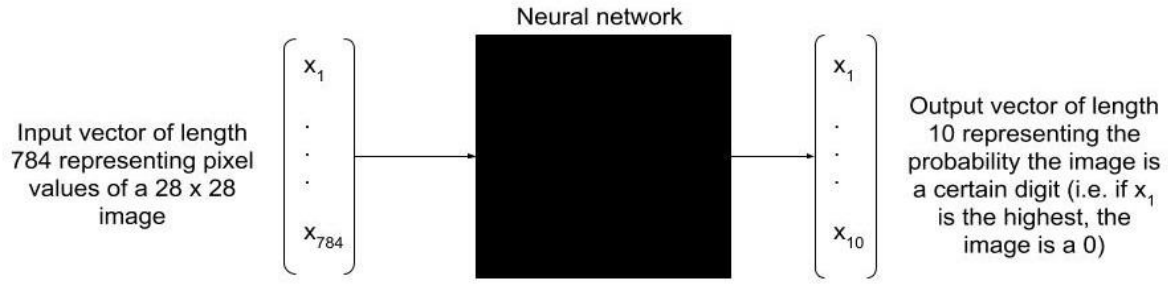

Figure 1: Neural Network as a Black Box – a neural network can be viewed abstractly as a black box function, taking in a vector of any dimension and outputting a different vector of any other dimension

The Artificial Neuron

The foundational building block of a neural network is the artificial neuron, also simply called the neuron. A single neuron will take in a vector of any length and compute its inner product with some vector of weights. The inner product yields a scalar value and then some constant bias is added to that scalar to potentially shift the function, allowing for more precision in function approximation. Then, an activation function is applied to that result to achieve another scalar. Activation functions can be any real-valued function, and they often provide non-linearity which expands the scope of functions the network can approximate. Figure 2 demonstrates this operation—given a 3-dimensional vector (x_1, x_2, x_3) and an activation function *f*, the output of the neuron will be

$$f\big((x_1, x_2, x_3) \cdot (w_1, w_2, w_3) + b\big),$$

where the weights w_1, w_2, w_3 and the bias *b* are "trainable" parameters whose values are obtained through optimization algorithms to find the ideal values (see the "Backpropagation" section). The true power of a neural network comes from the ability to link and stack multiple neurons together to create deep networks that model complex functions.

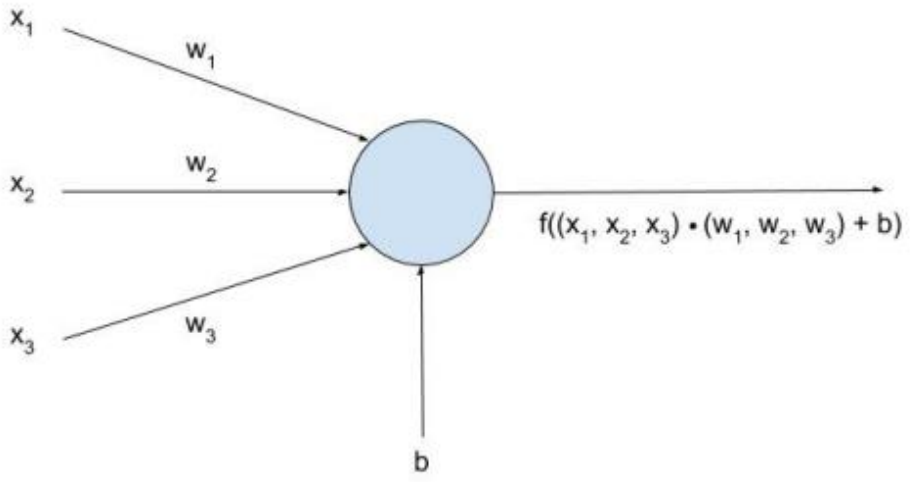

Figure 2: Artificial Neuron – a three-dimensional input vector passed into the neuron gets multiplied by the weights, added to a bias, and applies an activation function to output a real number

Hidden Layers

Multiple neurons can be stacked together in what is known as a hidden layer. Each neuron in the hidden layer has its own set of weights associated with each component of the input vector. For an input vector of dimension *n* and a hidden layer with *m* neurons, there will be *n* x *m* weight parameters. Hidden layers can be chained together sequentially, with the output of one layer serving as the input of the next layer. If one layer contains *m* neurons, its output will be *m* scalar values which can be treated as a vector for the next layer, with each neuron in this next layer having *m* weight parameters. Any number of hidden layers can be used if the vector dimensions match, but as more layers are added, the model's complexity increases

exponentially and will take increasingly longer to train and perform computations. Figure 3 provides a visual representation of two hidden layers and the corresponding weights between all neurons.

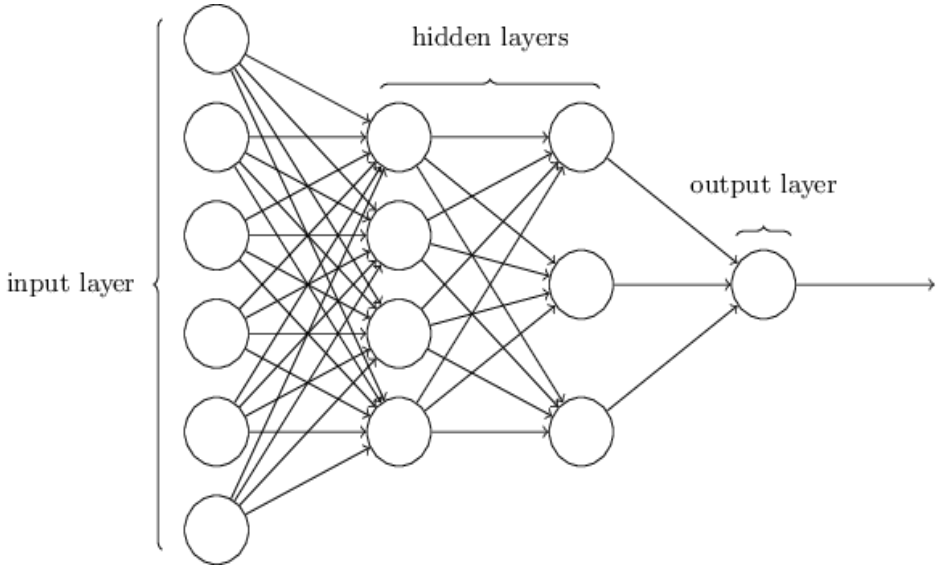

Figure 3: Neural Network with Two Hidden Layers – this representation of a neural network demonstrates how all the neurons and weights, which are represented as arrows, are connected

Forward Propagation

The actual operation of a neural network is to take in a vector and apply a series of vector operations to output a different vector which can be interpreted in various ways depending on the application. This process of applying vector operations is called forward propagation. Multiple vectors can be fed into the network at once by combining them into a single matrix, but then the weight vectors must also be augmented into matrices as well to allow for the matrix multiplication dimensions to match. For instance, each row of the input matrix would be a unique input vector, and then the weight vector at each hidden layer must be made into a matrix, with each column being the original weight vector. Feeding multiple input vectors at once as a matrix is preferred to one vector at a time because matrix operations have been optimized on computers, allowing more computations to be completed in a given timeframe than by using a single vector each time.

Gradient Descent and Backward Propagation

The ideal weight parameters (i.e. parameters that most accurately approximate the function) are calculated through the gradient descent algorithm. Gradient descent makes small updates to the weight vectors until the neural network achieves a desired accuracy. In order to optimize the weights, a loss function must be defined. Many loss functions exist, but a common one is taking the square of the L_2 norm of the difference between the intended output and the actual output. For instance, if the actual output was a vector (0.5, 0.5, 0) and the intended output was a vector (1, 0, 0), then the L_2 squared loss would be $(1 – 0.5)^2 + (0 – 0.5)^2 + (0 – 0)^2 = 0.5$. As an

equation, if y is the intended output vector and p is the actual output vector, then the L_2 squared loss is

$$L = \|y - p\|^2.$$

Gradient descent then will update the weights matrix of hidden layer i by the update rule

$$W_i = W_i - \eta \nabla L ,$$

where ∇L is the gradient of the loss function L with respect to W_i, and η is called the learning rate. Other papers discuss methods for choosing efficient learning rates that converge to optimal solutions quickly, so that information is omitted here.

To improve the efficiency of this algorithm, a common approach is to add the bias parameter into the weight vector, but this will require the data inputs to include an extra feature as well (a value of 1 is used) to have the dimensions match. The output p is a function of the weights at every layer in the network, so when taking the gradient of the loss, the chain rule will require that the last hidden layer weights be updated first, and then the previous layer can be updated from those updated weights. This creates an update rule that propagates backward through the neural network, starting at the end. After the weights are updated, inputs can be run through the network again, with the loss being recalculated and weights being updated once again. This process can be repeated many times until the neural network appears to be converging to minimal loss.

Summary

Artificial neural networks are machine learning models that can be used to create or model complex functions. Numeric data in the form of vectors are fed into the neural network, and vectors (or possibly scalars) are the outputs which may be interpreted in various ways depending on the application. How the outputs are computed depends on weight parameters internal to the network. Ideal weight parameters that maximize the accuracy of outputs are learned through optimization algorithms such as gradient descent. Once ideal weights are found from your training data, the goal of the neural network is to give accurate results for new data that has not been seen yet. The ability of neural networks to approximate nearly any mathematical function allows them to be used in almost any application in which information can be expressed numerically.

References

Figure 1. Neural Network as a Black Box. J. Dukewich, 2020.

Figure 2. Artificial Neuron. J. Dukewich, 2020.

Figure 3. Neural Network with Two Hidden Layers. Reprinted from Neural Networks and Deep Learning, by M. A. Nielsen, 2015, Retrieved from http://neuralnetworksanddeeplearning.com/chap1.html.