On the Functionality of the Multilayer Perceptron Classifiers

Landon Buell

18 May 2020

Introduction

In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts published *A Logical Calculus of the Ideas Immanent in Nervous Activity* [6], which paved the way for the fields of computational neuroscience and early mathematical models of the brain. McCulloch and Pitts proposed that one could loosely mimic the behavior of a brain as a series of mathematical calculations. They realized that the smallest known element of the brain, a *neuron*, acts as a simple on/off switch, much like computers have been designed to. Furthermore, single neurons by themselves do very little, but instead it is the organization of these cells into collections that make up functional networks [1, 4, 6].

The mathematical model could be augmented by trading the simple Boolean (T/F) value, with a continuously ranging number referred to as an *activation*. By producing different types of artificial neurons and establishing a series of synapses, it then became apparent that different organization of these elements could produce networks of solving increasingly advanced problems [2]. To further expand upon this, in 1957, Frank Rosenblatt created a neural network architecture called a Multilayer Perceptron (MLP), which uses layers of interacting neurons to create a single mathematical model [1, 7]. This gives it a simple feed-forward design which has experimentally shown to perform well with tasks related to image and speech recognition [2, 5].

In general, the goal of an classification MLP is to produce some approximation function, F^* that maps an array or vector of inputs, $\vec{x}$ to a set of discrete classes: $0, 1, 2, ..., k-1$, using a set of parameters $\hat{\theta}$ [1, 2, 7]. For a *k-bins* classifier, we notate this as:

$$F^* : \{\vec{x}, \hat{\theta}\} \rightarrow \{0, 1, 2, ..., k-2, k-1\} \tag{1}$$

This function is in turn composed of layers of smaller repeating functions, which is what gives rise to the *network* concept of the model. It is the chains of these smaller functions that allow for the producing of the approximate function F^* to be created and solve a vast range of problems.

Architecture

To build a Multilayer Perceptron (MLP), artificial neurons need to be grouped into multiple structures, each called a *layer.* A single layer is modeled mathematically as a column vector, $\vec{x}$ by convention:

$$\vec{x}^{(l)} = \Big[x_0, x_1, x_2, ..., x_{n-2}, x_{n-1}\Big]^T \tag{2}$$

The superscript (l) is used to indicate the layer number of which this vector represents. Each element inside this vector is a neuron in the MLP model, and the value of that element is the neuron's *activation* [1]. In term of numerical handling, this structure is an array of floating-point numbers. A neural network can have any number of these layers in them, with any number of neurons in each layer. The number of these hidden layers is often referred to as the *network depth* and the number of neurons in each layer is often called the *neuron density* or *network width.*

In addition to these layers, called *hidden layers*, a neural network also requires an *input layer* and an *output layer*[2]. As the name implies, the input layer is where initial arrays of data are passed into the network. This array is often called a *feature vector* as each element is a feature derived from a potentially larger data set. Typically, this feature vector is denoted by $\vec{x}^{(0)}$ is also the $\vec{x}$ object in equation (1).

In a network with L layers, the output array is also a vector, $\vec{x}^{(L-1)}$ which encodes the final decision made by the network model. In a k-bins classifier, there are k neurons, each one corresponding to a different class. The neuron with the highest activation value corresponds to the networks prediction for what class the particular sample, $\vec{x}^{(0)}$ belongs to. If the output, $\vec{x}^{(L-1)}$ is appropriately normalized, then then activations can be interpreted percentage probabilities of which class the sample belongs to:

$$P\big(\vec{x}^{(L-1)}\big) = \frac{1}{||\vec{x}^{(L-1)}||}\Big[x_0, x_1, x_2, ..., x_{k-2}, x_{k-1}\Big]^T \tag{3}$$

In this network of L layers, there are $L-2$ hidden layers whose activations are given by the entries in $\vec{x}^{(1)}$, $\vec{x}^{(2)}$, ... , $\vec{x}^{(L-3)}$, $\vec{x}^{(L-2)}$. And again, the input layers and output layers are given by $\vec{x}^{(0)}$ and $\vec{x}^{(L-1)}$ respectively. Throughout the duration of the model's construction, training and evaluation, these objects remain as column vectors / arrays of real floating point values.

Feed-Forward System

Back Propagation System

Conclusion

References

[1] Géron Aurélien. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly, 2017.

[2] Goodfellow, Ian, et al.*Deep Learning*. MIT Press, 2017.

[3] James, Gareth, et al. *An Introduction to Statistical Learning with Applications in R*. Springer, 2017.

[4] Levine, Daniel S. *Introduction to Neural and Cognitive Modeling*. 3rd ed., Routledge, 2019.

[5] Loy, James , *Neural Network Projects with Python*. Packt Publishing, 2019

[6] McCulloch, Warren S., and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, 1943, pp. 115–133.

[7] Petrik, Marek. "Introduction to Deep Learning." Machine Learning. 20 April. 2020, Durham, New Hampshire.