1 The Neural Network

1.1 An Introduction to Neural Networks

In the present age, some problems have shown themselves to be increasingly difficult to solve through conventional computer programs. The challenge arose as to how to build some sort of computer program that could function at a level above general procedural or explicit instructional rules. Rather than *hard-coding* a set of conditions or parameters for an algorithm, we seek an architecture that allows for a computer to *learn* and change and update itself as it is presented with more data. Such an algorithm exists in the form of a *Neural Network* [2, 3, 6].

Former YouTube video Classification team lead, and current Machine Learning consultant, Aurelien Geron writes [1]:

> Birds inspired us to fly, burdock plants inspired velcro and nature has inspired many other inventions. It seems only logical, then, to look to the brain's architecture for inspiration on how to build an intelligent machine.

The result of such an analogy is a computer program that is structured like the brain. Typically, a computer program is simply a set of mathematical or logical instructions given to a computer to execute. In 1943, Neuroscientist Warren McCulloch and Mathematician Walter Pitts published a paper [9] which began to pave the way for developing such a mathematical model of the brain [1, 3, 8].

1.2 The Structure of a Neural Network

A Neural Network is simply a model of a mathematical function, composed of several smaller mathematical functions called *layers* [1, 8]. Each layer represents an operation that takes some real input, typically an array of real double-precision floating-point numbers, and returns a modified array of new double-precision floating-point numbers. The exact nature of this operation can be very different depending on the layer type or where it sits within the network. It is this process of transforming inputs successively in a particular order until an output is attained [1, 8]. This output encodes the models final "decision" given a unique input.

A network model that contains L unique layers is said to be an L-Layer Neural Network that are usually index with a superscipt, 0 through $L-1$. Layer 0 is said to be the *input layer* and layer $L-1$ is said to be the *output layer*. The function that represents a layer (l) is given by

$$f^{(l)} : x \in \mathbb{R} \to y \in \mathbb{R} \tag{1}$$

The value of x can also be index by layer, we call the array $x^{(l)}$ the *activations* of layer l.

The the model is recursive by nature, the activations from one layer, $l-1$, are used to directly produce the activations of the next successive layer l. Thus eqn. (1) can be alternatively written as:

$$f^{(l)} : x^{(l-1)} \in \mathbb{R} \rightarrow x^{(l)} \in \mathbb{R} \tag{2}$$

The array of activations, $x^{(0)}$, is the raw input given the neural network, most commonly called *features*. Conversely, the activations $x^{(L-1)}$ are commonly called the network *output* [1, 4, 8].

1.3 Layers Used in Classification Neural Network

As stated previously, a neural network is composed of a series of functions that are called successively to transform features (an input) into a prediction (an output). As shown in eqn. (2), each function feeds directly into the next as to form a sort of computational graph [3].

Typically, a layer function can be divided into two portions: (i.) a Linear transformation, with a bias addition, and (ii.) an element-wise activation transformation. Many feed-forward network layers follow this structure. Step (i.) is usually in the form of a matrix-vector equation:

$$z^{(l)} = W^{(l)} x^{(l-1)} + b^{(l)} \tag{3}$$

Where $W^{(l)}$ is the *weighting-matrix* for layer l, $b^{(l)}$ is the *bias-vector* for layer l, $z^{(l)}$ are the *linear-activations* for layer l and $x^{(l-1)}$ is the final activations for layer $l-1$. Step (ii.) is usually given by some *activation function*:

$$x^{(l)} = \sigma^{(l)}\left[z^{(l)}\right] \tag{4}$$

Where $x^{(l)}$ is the final activations for layer l and $z^{(l)}$ is given in equation (3). $\sigma^{(l)}$ is some activation function, which is often use to enable the modeling of more complex-decision boundaries.

Below, we discuss and describe the types of layer functions that are used to produce the classification model in this project.

1.3.1 Dense Layer

The Linear Dense Layer, often just called a *Dense* Layer for short, was one of the earliest function types used in artificial neural network models. A dense layer is simply composed of a layer of *artifical neurons*, each of which holds a numerical value within it, called the *activation* of that neuron. This idea was developed back from McCulloch and Pitts' work [9], and was expanded upon by Frank Rosenblatt in 1957 [1].

We model a layer of neurons as a vector of floating-point numbers. Typically, it is required that the array be one-dimensional. Suppose a layer (l) contains n artificial neurons. We denote the array that hold those activations as $x^{(l)}$ and is given by:

$$\vec{x}^{(l)} = \Big[x_0, x_1, x_2, \ldots, x_{n-2}, x_{n-1}\Big]^T \tag{5}$$

The activation of each entry is given by a linear-combination of activations from the previous layer, as outlined in eqn.(3) and eqn.(4).

Suppose the layer $l-1$ contains m neurons. Then the weighting-matrix, $W^{(l)}$ has shape $m \times n$, the bias-vector $b^{(l)}$ has shape $m \times 1$. Thus for a dense layer l, the exact values of each activation is given by [1, 8]

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}^{(l)} = \sigma^{(l)} \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,m-1} \\ w_{1,0} & w_{1,1} & \dots & w_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n-1,0} & w_{n-1,1} & \dots & w_{n-1,m-1} \end{bmatrix}^{(l)} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}^{(l-1)} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{bmatrix}^{(l)} \right) \tag{6}$$

or more compactly:

$$x^{(l)} = \sigma^{(l)} \Big(W^{(l)} x^{(l-1)} + b^{(l)} \Big) \tag{7}$$

Eqn. (7) is generally referred to as the *dense layer feed-forward equation* [3].

1.3.2 2-Dimensional Convolution Layer

1.3.3 2-Dimensional Maximum Pooling Layer

1.3.4 1-Dimensional Flattening Layer

A flattening layer is used to compress an array with two or more dimensions down into a single dimension. For a flattening layer l, multidimensional activations in layer $l-1$ are compressed down into a single dimensional array. We can use function notation to express this as:

$$f^{(l)} : x^{(l-1)} \in \mathbb{R}^{(M \times N \times \dots)} \rightarrow x^{(l)} \in \mathbb{R}^{(MN \dots \times 1)} \tag{8}$$

The numerical value of each activation is left unchanged. For a layer with activation shape$N \times M$, the resulting activations are reshaped into $NM \times 1$ as shown in eqn (8).

Flattening Layer are most commonly used to prepare activations for entry into dense layer or series of dense layers. For example, 2D or 3D images are typically processed with 2D convolution, which may output a 2D or 3D array of activations. Those values are then flattened to 1 dimensions, which can then be passed into dense layers for further processing.

1.3.5 1-Dimensional Concatenation Layer

1.4 Training a Neural Network

1.5 Chosen Model Architecture

References

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

[2] Geron, Aurelien. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd ed., O'Reilly, 2019.

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

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

[5] Khan, M. Kashif Saeed, and Wasfi G. Al-Khatib. "Machine-Learning Based Classification of Speech and Music." Multimedia Systems, vol. 12, no. 1, 2006, pp. 55–67., doi:10.1007/s00530-006-0034-0.

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

[7] Liu, Zhu, et al. "Audio Feature Extraction and Analysis for Scene Segmentation and Classification." Journal of VLSI Signal Processing, vol. 20, 1998, pp. 61–79.

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

[9] 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.

[10] Mierswa, Ingo, and Katharina Morik. "Automatic Feature Extraction for Classifying Audio Data." *Machine Learning*, vol. 58, no. 2-3, 2005, pp. 127–149., doi:10.1007/s10994-005-5824-7.

[11] Peatross, Justin, and Michael Ware. *Physics of Light and Optics.* Brigham Young University, Department of Physics, 2015.

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

[13] Short, K. and Garcia R.A. 2006. "Signal Analysis Using the Complex Spectral Phase Evolution (CSPE) Method." AES: *Audio Engineering Society Convention Paper*.

[14] Virtanen, Tuomas, et al. *Computational Analysis of Sound Scenes and Events.* Springer, 2018.

[15] Zhang, Tong, and C.-C. Jay Kuo. "Content-Based Classification and Retrieval of Audio." *Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, 2 Oct. 1998, pp. 432–443., doi:10.1117/12.325703.