Sigmoid Neuron B+

W. Max Lees

June 10, 2015

1 Introduction

Generally, sigmoid neuron output is based on a simple logistical equation. My goal is to take this simple equation and expand it. Instead of using a single bias for the output to the entire next layer of neurons, the B+ neuron will have a vector of biases. This will allow the neuron to not only change it's weight per input, but change it's bias per output.

2 Logistic Equation

Let's start by assuming that both σ (the output of the neuron) and b (the bias for each output) are vectors of size n, where n is the number of neurons in the next layer.

$$\text{let } \sigma, b \in \Re^n \tag{1}$$

The resulting equation would have much the same shape as the usual sigmoid neuron output equation.

$$\sigma = \frac{1}{1 + e^{-(w \cdot x) - b}} \tag{2}$$

It is important to remember that both σ and b are vectors. So,

$$\sigma_j = \frac{1}{1 + e^{-(w \cdot x) - b_j}} \tag{3}$$

But this equation gives us a bit of a problem for our computing speeds. For every output, σ must be calculated each time. Let's see if we can't improve on this slightly. The dot product between w and x will generate a constant which can be calculated once and then used each time. So,

$$\text{let } c = (w \cdot x) \tag{4}$$

That single computation will save us some calculations per output. The resulting equation looks like this.

$$\sigma = \frac{1}{1 + e^{-c-b}} \quad (5)$$

For the sake of later improvements, we can make one additional change to the equation:

$$\sigma = \frac{1}{1 + e^{-c}e^{-b}} \quad (6)$$

Again, because c only needs to be calculated once, we can also raise its exponent once. We'll continue to call the value c because it is still a constant.

$$\sigma = \frac{1}{1 + ce^{-b}} \quad (7)$$

Before we make our next change, it might be helpful to examine the bias b in the normal logistic function. This number is chosen at random and then adjusted by the learning algorithm over many iterations. The important idea to note at this point, is that b is not generated by some real world number, it is instead chosen. As a result, we can save ourselves even more calculation by building in our exponent. That is, instead of choosing our b values, we choose values for e^b. Generally, in normal Sigmoid Neurons, $b \in \mathbb{Z}, -10 < b < 0$. So for our new Sigmoid Neuron B+, let's create a new vector for holding an adjusted b vector. Let's call it b'. $b' \in \mathbb{Z}, b'_j = e^{b_j}$. But, instead of calculating b', we just ensure that the value fits within the bound already set by its definition. If we build in our exponent, we no longer have to calculate one for each b value. As a result, here is our equation:

$$\sigma = \frac{1}{1 + cb'} \quad (8)$$

With this equation, we have a fairly fast calculation per neuron compared to the original equation. The value of c is calculated once as $c = e^{-(w \cdot x)}$. b' requires no calculations but instead has all of its calculations built in by the numbers the program chooses. We are left with n multiplications for our cb', n additions with our $1 + cb'$, and n divisions with the final $\frac{1}{1+cb'}$. These can all be performed at once to increase speeds based on caching.