Introduction to Neural Networks

June 21, 2020

Neural Networks try to resemble the way humans make decisions and the way neurons in the human brain take in data and act on input.

Introduction

A neuron usually has a couple parts to it. There are the **dendrites**, which are in charge of receiving incoming signals from other neurons. The signals then travel through the cell body and the **axons** all the way to the connections at the end of the axons. These neurons are all connected to many other neurons each, so any attempt to model these connections will necessarily use up a lot of connections.

In computer science, we hae something resembling a neuron, except that the way we model decisions is a bit different.

Neural networks are usually represented by a connected graph of neurons, usually organized in layers. There are a couple types:

1. **Feedforward**: In these types, the signals only flow in one direction.
2. **Recurrent**: In recurrent neural networks, there is feed forward functionality, but there can also be feedback to the neurons.

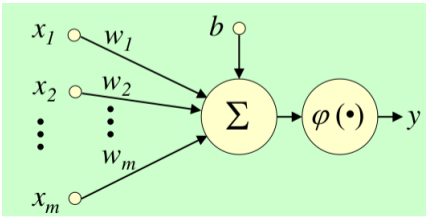

Figure 1: We take a set of inputs $x_1 \dots x_n$ and pass each of these through a corresponding **weight function** $w_1 \dots w_n$. In the actual node body, we take the sum of the weight functions connecting the current neuron to the ones in the previous layer. This sum Σ is then passed through the final **activation function** φ, which gives the final output for this neuron.

In a neural network, we would usually have k output layers, which might represent the confidence with which we classify or predict a certain value for our model. Let the final value of our model ve $y(x)$ for some input vector x. The error value for a neural network would then be

$$e = d - y$$

which we of course want to minimize.

Perceptron Algorithm

The perceptron algorithm is an algorithm which can learn the classification from the data. How does it do this? It only updates the set of weights when it makes a mistake in classifying or regressing(?) the testing data.

Originally, with a single neuron, the algorithm could only handle the cases that were **linearly separable**. For example, training a model to learn the behavior of the XOR function is not possible with only a single neuron, again since the training data is not linearly separable. However, with a simple two-layer network we can easily model the behavior of XOR.

So, for a single neuron, updating the values of **w**