Neural Networks

November 9, 2015

Abstract

In this short paper we will discuss the fundamentals of neural networks and their implementation in detail. We will give a general overview of how neural networks work, discuss calculation of the gradient and implementation of back-propagation, and test our results on some real MNIST code.

Neural networks have been around for at least a few decades, but only recently have them become popular as a method for learning parameters that can correctly translate an input into an output. This is because of increased computational power, a greater availability of training data, as well as the fact that more complex models, like deep neural nets, are actually easy to train - the same back-propagation that works to update normal neural networks works just as well for multiple hidden layers. Figure 1 shows a basic neural network. The x on the left is the input,

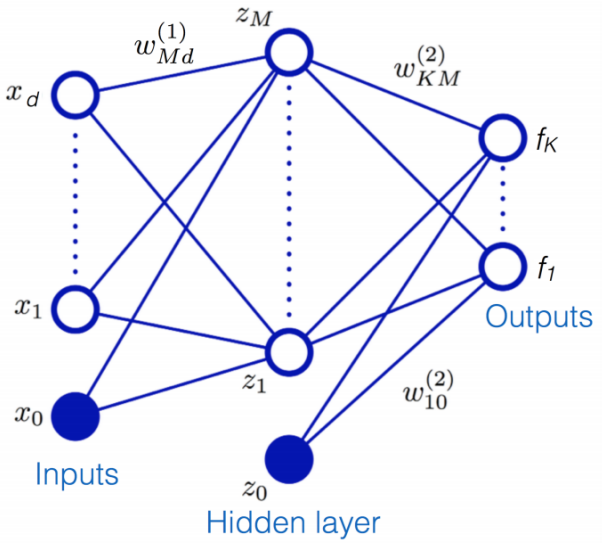

Figure 1: 1-hidden layer neural networ taken from Bishop

with each of the d dimensions acting as a separate node. These are then multiplied by the appropriate weights, added to a *bias* (representing x_0 here), in order to obtain the *activations* a_j for $1 \leq j \leq M$. These activations are then put through some non-linear map, in this case the sigmoid function, to obtain the values at the first hidden layer z_j. Mathematically, this looks like:

$$a_j^{(1)} = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

$$z_j = g(a_j) = \frac{1}{1 + e^{-a_j}}$$