Optimization Methods for Deep Feed-Forward Neural Networks

Landon Buell

25 June 2020

1 Introduction

1.1 Deep Feed-Forward Neural Network Structure

A Deep-Feed forward neural network is a class of architectures built around layers of operations that are called in sequence. Each layer, (l) takes some input object $x^{(l-1)}$, applies some operation, $f^{(l)}$, and produces some output $x^{(l)}$, where the super-script indicates a layer index. The object $x^{(l)}$ then becomes the input for the next layer of the network. A model with L layers requires an input object $x^{(0)}$, which is then transformed by $L-1$ layers to produce object $x^{(L-1)}$, which we refer to as the out of the network.

Each layer of the network takes the form of a function-like object f that takes an input $x \in \mathbb{R}^N$ and transforms it into an output $y \in \mathbb{R}^M$. Typically, f can take the form of a matrix-vector equation, a k-dimensional convolution operation, a pooling operation, a non-linear activation function, or any other general transformation. In each case, there as a series of parameters associated with the function f. In the case of a matrix-vector equation, there are the elements inside each matrix and bias vector, and in the case of convolution, there are the weighting values within the kernel itself.

1.2 Optimization Motivation

Ultimately any output is function of the input, and the parameters in each layer. By convention, these parameters can be concatenated into a single object called θ. For smaller neural networks, θ may only have several hundred elements, but for increasingly complex models, there can upwards of several million elements within the object. The goal of the network then is to find a particular set of elements $\{\theta_0, \theta_1, ...\}$ that allow a network to produce a desirable out based on that input. To do this, we require a set of samples $X = \{x_0, x_1, x_2, ...\}$ and a set of corresponding labels $Y = \{y_0, y_1, y_2, ...\}$, such that element x_i produces y_i, when passed into the network model.

In reality, a network will not produce output y_i from samples x_i exactly, but will instead produce an *approximation* of y_i which we denote as y_i^*. We can use a function J called the *objective function* which compares the expected output, y_i, to the actual output, y_i^* and produces a numerical score ranking how much the two values differ from each other. This score is called the *cost* of the sample, lower numbers being favorable, showing a low difference or low cost of the sample.

The value of the cost function is then directly dependent on the expected output, y_i and the given output y_i^*. The cost of a single sample can be shown as $J = J(y_i, y_i^*)$. However, the output, is implicitly dependent on the the network input x_i and the parameters in each layer, given by θ. Since the latter two are fixed objects, we can then only adjust the elements in the object θ. We do so to allow for increasingly lower values of the objective function, which indicates that on average, the network model is producing more and more accurate predictions. This process is called *optimization*.