Introduction to Stochastic Gradient Descent

Explained through Neural Networks

Landon Buell

27 January 2020

1 Introduction

Gradient Descent algorithms lie at the heart of several machine learning algorithms [3]. In general, the goal of the algorithm is find a set of parameters that minimize the value of a particular function, which we call the *Cost Function* or *Objective Function* [1]. To do this, we apply a procedure from multivariate calculus that repeats a procedure until a local minimum of that particular function is found. Note that while we ideally want to find a global minimum (the minimum of the whole function), this is computationally unrealistic, so local minima of the function are used in it's place.

Suppose we have some scalar defined function, f of n independent variables. We notate this as:

$$f = f(x_0, x_1, ..., x_{n-2}, x_{n-1}) \tag{1}$$

The gradient of the function, returns a vector, with each element given by the partial derivative of f with respect to the i-th variable:

$$\nabla\big[f(x_0, x_1, ..., x_{n-2}, x_{n-1})\big] = \Big[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_{n_2}}, \frac{\partial f}{\partial x_{n-1}}\Big] \tag{2}$$

Geometrically, the gradient provides a vector that points in the direction that causes the value of f to increase the fastest. In a two dimensional function, where the output is the 3rd dimension, this gives the direction of steepest ascent. Note that if the function of of n variables, then this operation take place in n-dimensional vector space.

The general procedure of the Gradient Descent Algorithm is:

1. Pick a 'starting point', p_0 in the n-d space. This is done by evaluating the function f for each of it's n input variables.

2. Compute the gradient of f, eq. (2) at the point p_0. Multiply the gradient by -1. This is the direction of steepest *descent*.

3. Follow the negative gradient to reach a new point p_1.

4. Repeat steps 2 and 3 until the value of $-\nabla f$ returns 0. This indicates that a local minimum, or a saddle point has been found.

Some gradient descent algorithms may repeat this whole procedure with a series of initial points, each time tracking the minimum and corresponding parameters. This way, each time the algorithm is repeated, a new minimum is attained, which allows for a greater chance to find a successively lower value. While it seems appealing at first thought, it is work noting that this whole algorithm as outlined is *very* computationally expensive.

2 Considerations and Conventions

2.1 Structure

It is important to understand the architecture of a convolution neural network (CNN) at a very basic level. In it's simplest form as well will describe, a neural network is a collection of layers of functions (often called *nodes* or *neurons*) [3]. The exact amount of layers in a network depends on and the exact number of neurons in each layer depends on the exact type of task that the network seeks to accomplish. Currently, there is no formal rule to outline this exactly.

The entry point of a neural network, called *Layer 0* is the set of functions (or values) that receives an initial piece of data. The number of these inputs neurons in this layer corresponds to the number of *features* in the base data set. Thus if a sample of data has n_0 features, then there are n_0 neurons in this layer 0 of the neural network [2].

To move to the next layer, an operation is applied to all of the values in the previous layer. We model this with standard matrix multiplication. notation conventions for this will outlined in the next section. After each matrix multiplication is applied, the input is effectively transformed into the next layer of the network. It is important to know that although layer 0 has n_0 neurons, any other layer may have a different number of neurons in it, This difference is handled by the dimension of the matrix that allows for the transformation between adjacent layers.

In a network with L layers, there are $L-2$ matrix multiplications required (due to 0 - indexing). Once the $L-2$ matrix has been applied to layer $L-2$, the resultant layer, $L-1$, is effectively the output of the network. In the case of a CNN being used for a classification problem, the number of nodes in the final layer of the network, corresponds to the number out target classes. Thus if we built a *K-Folds* classifier, the output layer of the CNN would have $n_{(l-1)} = K$ nodes.

2.2 Notation

The mathematical description of stochastic gradient descent (SGD) classifiers is largely based in linear algebra [3]. This means a great deal of matrix vector indexing is required to fully describe the procedure. For this work, I will be using a standard of 0 - indexing all objects. I outline a notional convention for this work:

- $\vec{x}_i^l$ is the i-th entry in the vector x in the l-th layer of a network. The vector that describes layer l of the network has n_l rows and 1 column.
- n_l is the number of neurons (also called nodes of functions) in the l-th layer.
- $W_{i,j}^l$ is the entry in the i-th row, and the j-th column of matrix W that operates on the l-th layer of a network. The matrix that operates on layer l has $n_{(l+1)}$ rows and n_l columns

For this

References

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

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

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

[4] Petrik, Marek. "Introduction to Machine Learning." Machine Learning. 22 Jan. 2020, Durham, New Hampshire.