Backpropagation Through Weights and Biases

John Purcell

December 11, 2021

1 Input, Weights and Loss

Suppose we transform an input matrix via weights and biases. We can write this as the following equation, where the columns in the bias matrix B are all identical, since we add the same biases for all columns of input.

$$Y = WX + B$$

X: the input matrix

Y: the output matrix

W: the weights matrix

B: the bias matrix

We can then further transform Y into a loss matrix λ

$$\lambda = f(Y)$$

The loss matrix λ contains one value for every column in the input matrix. It is produced by applying a function $f()$ to Y, where $f()$ may consist of multiple subsequent stages of a neural network (activation functions, more weights and biases, etc.)

We can combine these two equations as follows.

$$\lambda = f(WX + B)$$

We will consider equations for only one column of input, since it simplifies the resulting mathematics. A column of input does not influence the loss corresponding to a different column of input. Considering multiple columns of input would therefore simply require us to use additional subscripts denoting the input/output column being considered.

The above equations then become:

$$\vec{y} = W\vec{x} + \vec{b}$$

$$\lambda = f(\vec{y}) = f(W\vec{x} + \vec{b})$$

where

λ is a single floating point value

$\vec{y}$ is a column vector of output values

W is a matrix of weights

$\vec{x}$ is a column vector of input values

$\vec{b}$ is a column vector of biases

Note that

A particular value in the weights matrix may be denoted by w_{ij}

A particular value (at a particular row i) in the input vector may be denoted simply by x_i

Similarly, a value in the bias vector may be denoted by b_i

A value in the output vector may be denoted by y_i

2 Applying the Chain Rule

Given

$$\frac{\partial \lambda}{\partial y_i}$$

we want to calculate:

$$\frac{\partial \lambda}{\partial x_i}$$

For the purposes of applying the chain rule, we may think of $\vec{x}$ as a function of $\vec{y}$.

Then we can write the following:

$$\frac{\partial \lambda}{\partial x_i} = \sum_n \frac{\partial \lambda}{\partial y_n} \frac{\partial y_n}{\partial x_i} \tag{1}$$

For a neural network, the term y_n is the output of the nth neuron (in some particular layer under consideration). It is calculated like this:

$$y_n = \left(\sum_m w_{nm} x_m \right) + b_n$$

When we differentiate this with respect to x_i, any bias term added makes no difference to the resulting rate of change. The rate of change of y_n with respect to x_i is simply the value of the weight that multiplies x_i.

$$\frac{\partial y_n}{\partial x_i} = w_{ni}$$

Equation 1 becomes:

$$\frac{\partial \lambda}{\partial x_i} = \sum_n \frac{\partial \lambda}{\partial y_n} w_{ni} \tag{2}$$

We will refer to the rates of change of the loss with respect to the elements of the input vector $\vec{x}$ and the elements of the output vector $\vec{y}$ as "error" vectors, denoted using the letter δ ("delta") as follows:

$$\delta_i^{in} = \frac{\partial \lambda}{\partial x_i} \tag{3}$$

$$\delta_i^{out} = \frac{\partial \lambda}{\partial y_i} \tag{4}$$

Equation 2 can then be written as:

$$\delta^{\vec{in}} = W^T \delta^{\vec{out}} \tag{5}$$