mathwork

Adam Ibrahim

September 2025

1 Introduction

2 MLOPS

Forward propagation seems to work by just executing the neural network with random weights and biases

$$Z = WX + b$$

Where Z is the neuron W is the weight matrix, X is the input vector, and b is the bias, for binary outputs eg (Right or Wrong) you use the sigmoid function to use as the activation function. ReLU is defined as

$$\text{ReLU}(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

This is also the sigmoid function which you can use for activation. In the notes I'll be using it , but in practice I'll be using ReLU. Sigmoid is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

There's the tanh function which is short for tan hyperbolic.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Now this comes at the cost at computing 4 exponentials. ReLU seems to be the cheapest out of them all.

ReLU seems to be the cheapest and seems to get the job done so I'll go with that.

I wanted to know how to activate a neuron and with this I'll be using the sigmoid for the math but it doesn't matter that much

$$\mathbf{W} = \begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{pmatrix}$$

So im going to let W be the weights of our neural network of general size and I'll let I as in input be the input vector which should look like

$$I = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix}$$

with these computing a vector matrix product should be relative ease with the matrix ops file having that method. Now I'll let another vector with the biases

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Now this is where the *magic* comes in the actual ML

$$a_0^{(1)} = \sigma(\mathbf{W}I + B)$$

$$a_0^{(1)} = \sigma(w_{0,0}a_1 + w_{0,1}a_2 + \cdots + w_{0,n}a_n + b_1)$$

this is for activating ONE neuron if we have hundreds of neurons in our hidden layers then it's going to redo this computation maybe thousands of times. Now we need a cost function or a boolean output

This is going to be in the form of right and wrong and what is the correct answer is. So let $C(x)$ be our cost function

$$C(x) = \Sigma(\text{ResultVector}_n - \text{AnswerVector}_n)^2$$

The result vector is the results you get from the output layer and the answervector should be what the output layer should be so it should look like a bunch of zeros then a 1 and the resultvector should look like a bunch of numbers from 0-1 ergo confidence in the output

Large values in $C(x)$ is BAD it means that the NN doesn't know anything and it's garbage and we gotta get it low as possible.

3 MatrixOps

A matrix is a 2d array in cs or a 2d vector a matrix is denoted by uppercase letters so "A" is a matrix but "a" is not a matrix. You could also bold it to

emphazise it but it's up to you.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

This is a 3x3 matrix, you can also have non-square matrices like

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

this is a 2x3 matrix, you can also have a mxn matrix where m is the number of rows and n is the number of columns. You can also have a 1xn matrix which is a row vector or a mx1 matrix which is a column vector. You can also have a 1x1 matrix which is just a scalar. With this you can relate this to a system of equations like

$$f(x) = \begin{cases} 2x + 3y = 6 \\ 4x + 5y = 10 \end{cases}$$

This can be represented as a matrix equation like

$$\begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}$$

This is a coefficent matrix, where it's just the coefficents of the variables in the equations. You can also have an augmented matrix which is the coefficent matrix with the constants on the right side of the equations. This is called an augmented matrix because it's augmented with the constants.

$$\left(\begin{array}{cc|c} 2 & 3 & 6 \\ 4 & 5 & 10 \end{array}\right)$$

with these augmented matrices we can generalize any system of equations to an augmented or coefficent matrix. This is useful because we can use row operations to solve the system of equations. Row operations are just operations that we can do to the rows of the matrix to get a solution. You already used row operations in middle school when you did substitution and gaussian elimination A way to describe a solution for a matrix is to put it in row echelon form or reduced row echelon form. Row echelon form is when the leading coefficient of each row is to the right of the leading coefficient of the previous row. The leading coefficient is the first non-zero number in a row. It looks like

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 6 \\ 0 & 1 & 4 & 5 \end{array}\right)$$

This is in row echelon form

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & -14 \\ 0 & 1 & 0 & 5 \end{array}\right)$$

This is in reduced row echelon form

To get to row echelon form you can use the following row operations:

- Swap the positions of two rows (interchange)
- Multiply a row by a non-zero scalar (scaling)
- Add or subtract a multiple of one row to another row (replacement)

Matrix multiplication is multiplying two matrices (I know shocking)

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \end{pmatrix}$$

4 linear regression

MLR

$$(X^\top X)^{-1} X^\top Y$$

this is also the equation for a linear line

$$y = mx + b$$

$$m = \frac{n\sum y \sum x^2 - \sum y \sum xy}{n\sum x^2 - (\sum x)^2}$$

$$b = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$$