In order to find the derivative of the cost function with respect to w and b, where w is an array of weights and b is a scalar representing bias, a strategy called “Back-propagation” is used. We are given the cost function:

cost = np.squeeze( (-1. / num\_images) \* np.sum((Y\*np.log(A)+(1-Y)\*np.log(1-A)),axis=1) )

It is important to recall that “A” is referring to the activation calculated using the activation function, a sigmoid in this case, where and . Through this activation calculation, the terms “w” and “b” actually appear in the function. The derivative of the cost function is acquired via the chain rule, which connects the different inputs of the equation. Where C is cost, and , . The same strategy is used to find . Ultimately, in the code, the derivatives are:

dw = (1./num\_images)\*np.dot(X,((A-Y).T))

db = (1./num\_images)\*np.sum(A-Y,axis=1)