

$$\frac{\partial L}{\partial w_1} = (a - y) \cdot x_1$$

Here,  $(a - y) = \frac{\partial L}{\partial z}$

$$w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1}$$

Similarly, for all parameters

$$w_i = w_i - \alpha \frac{\partial L}{\partial w_i}$$

$i = 1, 2, \dots, m$   
 $m = \text{no. of parameters}$

$$b = b - \alpha \frac{\partial L}{\partial b}$$

where,  $\frac{\partial L}{\partial b} = (a - y)$

Figure 12: Gradient Descent part 2

## Python Implementation

```
def weightInitialization(n_features):
    w = np.zeros((1, n_features))
    b = 0
    return w, b

def sigmoid_activation(result):
    final_result = 1 / (1 + np.exp(-result))
    return final_result

def model_optimize(w, b, X, Y):
    m = X.shape[0]

    # Prediction
    final_result = sigmoid_activation(np.dot(w, X.T) + b)
    Y_T = Y.T
    cost = (-1/m) * (np.sum((Y_T * np.log(final_result)) + ((1 - Y_T) *
    (np.log(1 - final_result)))))
    #

    # Gradient calculation
```