Logistic Regression with a Neural Network mindset

Sunday, June 10, 2018
11:05 PM

Logistic Regression is a simple linear classifier. We can consider this as a simple neural network with only ONE neuron.

STEP 1: Load dataset + preprocessing (Flatten)
Shape of dataset matrix: (12288=64*64*3)

train_set_x_flatten shape	(12288, 209)
train_set_y shape	(1, 209)
test_set_x_flatten shape	(12288, 50)
test_set_y shape	(1, 50)

STEP2: Initialize parameters (w, b)

W	(dim, 1)	Matrix
b	0	Integer

STEP3: Implement propagate function
3.0 Notes

Shape of X	(num_of_dims, num_of_samples)
Shape of Y	(1, num_of_samples)

3.1 Forward propagate: predict X

- You get X
- You compute $A = \sigma(w^T X + b) = (a^{(0)}, a^{(1)}, ..., a^{(m-1)}, a^{(m)})$
- You calculate the cost function: $J = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(a^{(i)}) + (1-y^{(i)})\log(1-a^{(i)})$

3.2 Backward propagate: compute gradient of w and b

$$\frac{\partial J}{\partial w} = \frac{1}{m}X(A-Y)^T \qquad \frac{\partial J}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(a^{(i)} - y^{(i)})$$

3.3 Implementation
Ref: https://blog.csdn.net/u012609509/article/details/70230204

np.dot()	Standard matrix multiplication
np.multiply()	element-wise product (Multiply each element of two vectors/matrix respectively)
*	Same as np.multiply()

np.sum()	Sum of all elements

STEP4: Implement optimization function
Modify w, b by minimizing cost function
w = w - learning_rate*dw
b = b - learning_rate*db

STEP5: Now predict!

$$\hat{Y} = A = \sigma(w^T X + b)$$

Then convert the entries of A into 0 (if activation <= 0.5) or 1 (if activation > 0.5)

Shape of A	(num_of_dims, num_of_shapes)