deeplearning.ai

Basics of Neural Network Programming

Logistic Regression Gradient descent

using computation graph.

Logistic regression recap

Say x has x_1 & x_2
$\Rightarrow$ w has w_1 & w_2

$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

$z = w_1 x_1 + w_2 x_2 + b \Rightarrow \hat{y} = a = \sigma(z)$

Logistic regression derivatives

$$\left.\begin{array}{l} x_1 \\ w_1 \\ x_2 \\ w_2 \\ b \end{array}\right\} \longrightarrow \boxed{z = w_1 x_1 + w_2 x_2 + b} \longrightarrow \boxed{a = \sigma(z)} \longrightarrow \boxed{\mathcal{L}(a, y)}$$

$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w_1}$

$\frac{\partial z}{\partial w_1} = \frac{\partial (w_1 x_1 + w_2 x_2 + b)}{\partial w_1} = x_1$

// ly $\frac{\partial z}{\partial w_2} = x_2$, $\frac{\partial z}{\partial b} = 1$

$\Rightarrow$

$$\boxed{\begin{array}{ll} \frac{\partial L}{\partial w_1} = \underbrace{x_1 (a-y)}_{dw_1}, & \\ \frac{\partial L}{\partial w_2} = \underbrace{x_2 (a-y)}_{dw_2}, & \underbrace{\frac{\partial L}{\partial b} = (a-y)}_{db} \end{array}}$$

$\Rightarrow$ In gradient descent

$w_1 := w_1 - \alpha \cdot dw_1$

$w_2 := w_2 - \alpha \cdot dw_2$

$b := b - \alpha \cdot db$

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

$\hookrightarrow \frac{dL}{da} \cdot \frac{da}{dz}$

we know this

$\frac{da}{dz} = \frac{\partial \left(\frac{1}{1+e^{-z}}\right)}{\partial z}$

$= \frac{\partial \left((1+e^{-z})^{-1}\right)}{\partial z}$

$= -1 \cdot (1+e^{-z})^{-2} \cdot (e^{-z}) \cdot (-1)$

$= \frac{e^{-z}}{(1+e^{-z})^2}$

$= \left(\frac{1}{1+e^{-z}}\right) \cdot \left(\frac{e^{-z}}{1+e^{-z}}\right)$

$= a \cdot (1-a)$

$$\boxed{\begin{array}{l} \frac{\partial L(a,y)}{\partial a} = \frac{-y}{a} + \frac{1-y}{1-a} \\ \frac{\partial \left(-[y \log a + (1-y)\log(1-a)]\right)}{\partial a} \\ = -y/a + \frac{(1-y)\cancel{(-1)} \cdot \cancel{(-1)}}{(1-a)} \\ = \frac{-y}{a} + \frac{(1-y)}{(1-a)} \end{array}}$$

$\Rightarrow \left[\frac{-y}{a} + \frac{(1-y)}{(1-a)}\right] \cdot a(1-a) = \left[\frac{-y(1-a) + (1-y)a}{a\cancel{(1-a)}}\right] \cdot \cancel{a(1-a)} = -y + \cancel{ay} + a - \cancel{ay}$

$= a - y$

Now that we have $\frac{\partial L}{\partial z}$, we can find $\frac{\partial L}{\partial w}$, $\frac{\partial L}{\partial b}$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(a^{(i)}, y^{(i)})$

overall cost func

$a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$

for a single example

$(x^{(i)}, y^{(i)}) \rightarrow X \rightarrow x_1, x_2$

we had found

$dw_1^{(i)}, dw_2^{(i)}, db^{(i)}$ [last slide)

- What we really have to do now is to find $\frac{\partial J}{\partial w_1}$, ie, iterate over all m samples, find $dw_1^{(i)}$ for each sample & that becomes our gradient for w_1.

$$\frac{\partial J}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial Loss(a^{(i)}, y^{(i)})}{\partial w_1^{(i)}}$$

- once we have this, we can update w_1 for that iteration

$$w_1 := w_1 - \alpha \cdot \frac{\partial J}{\partial w_1}$$

But this is just w_1

we will have to do the same for w_2, b

Remember

- Loss func is on 1 sample "L"
- cost func "J" is on m samples

So we have,

$$\frac{\partial J(w,b)}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial L}{\partial w_1}(a^{(i)}, y^{(i)})$$

$$\frac{\partial J(w,b)}{\partial w_2} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial L}{\partial w_2}(a^{(i)}, y^{(i)})$$

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial L}{\partial b}(a^{(i)}, y^{(i)})$$

$$w_1 := w_1 - \alpha \cdot \frac{\partial J}{\partial w_1}$$

then $w_2 := w_2 - \alpha \cdot \frac{\partial J}{\partial w_2}$

$$b = b - \alpha \cdot \frac{\partial J}{\partial b}$$

& Repeat

Logistic regression on m examples

$J=0,\ dw_1=0,\ dw_2=0,\ db=0$

For $i=1$ to m

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J += -\left[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)}) \right]$ // cost func (m examples) = cost func (m examples) + loss func for $x^{(i)}, y^{(i)}$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 += x_1^{(i)} dz^{(i)}$

$dw_2 += x_2^{(i)} dz^{(i)}$

this is assuming X has only 2 features (x_1, x_2), if there are more, they come here

This is finding $\frac{\partial L}{\partial w_i}$ & adding it to ultimately find $\frac{\partial J}{\partial w_i}$

$db = dz^{(i)}$

end For

This part is calculating $\frac{1}{m}\sum_{i=1}^{m} \frac{\partial L}{\partial w_1}$, $\frac{1}{m}\sum \frac{\partial L}{\partial w_2}$, $\frac{1}{m}\sum \frac{\partial L}{\partial b}$

// Now that you have found total loss, total dw_1, dw_2, db, find avg ⇒ divide by m

$J = J/m$

$dw_1 = dw_1/m$

$dw_2 = dw_2/m$

$db = db/m$

outside For loop

// Now you can update the weights as you've found $\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \frac{\partial J}{\partial b}$

$w_1 = w_1 - \alpha \cdot dw_1$

$w_2 = w_2 - \alpha \cdot dw_2$

$b = b - \alpha \cdot db$

This was 1 iteration of updated weights, now go & find the new $\hat{y}$ with this & repeat 10K times till there is no change in the weights in this section