Neural Networks Basics

Semana 2

Alida Zárate 28/Sep/2022

deeplearning.ai

Basics of Neural Network Programming

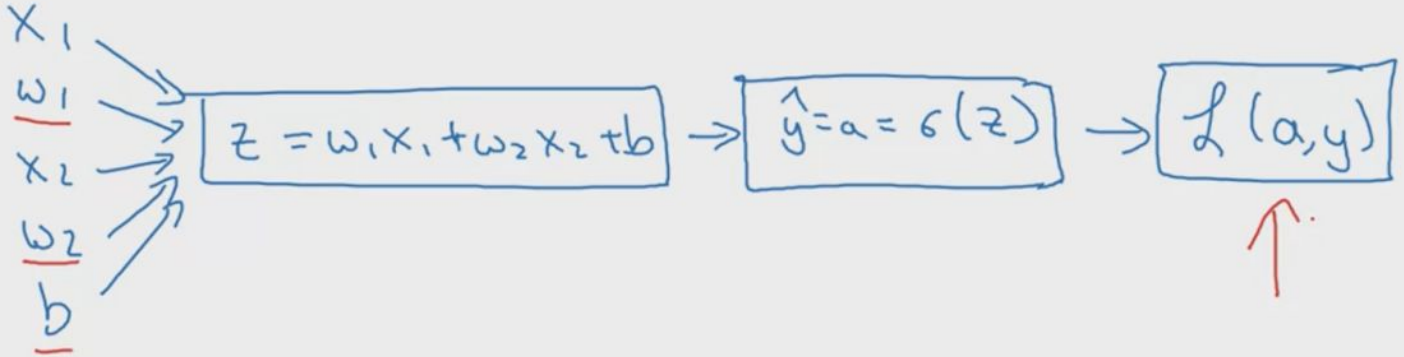
Logistic Regression Gradient descent

Logistic regression recap

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

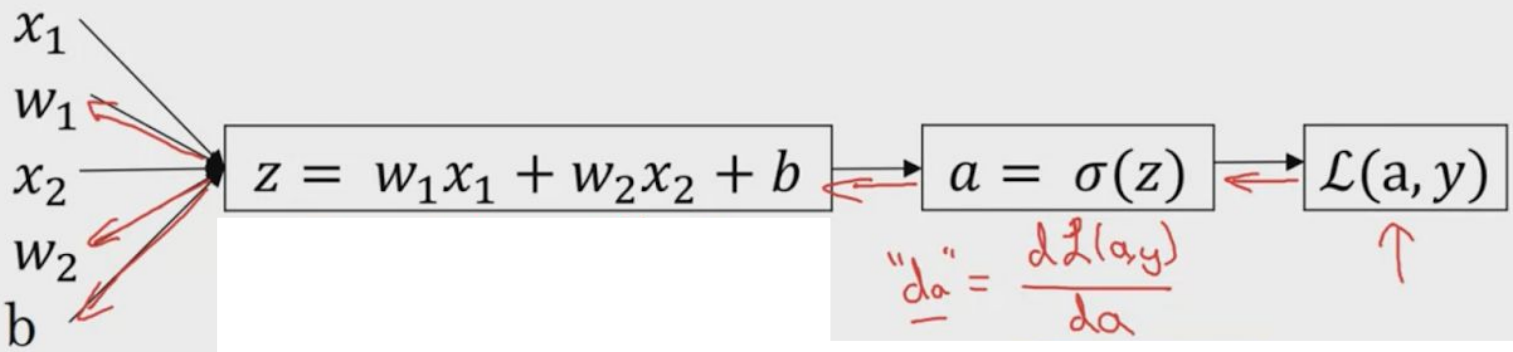
$$\hat{y} = a = \sigma(z)$$

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

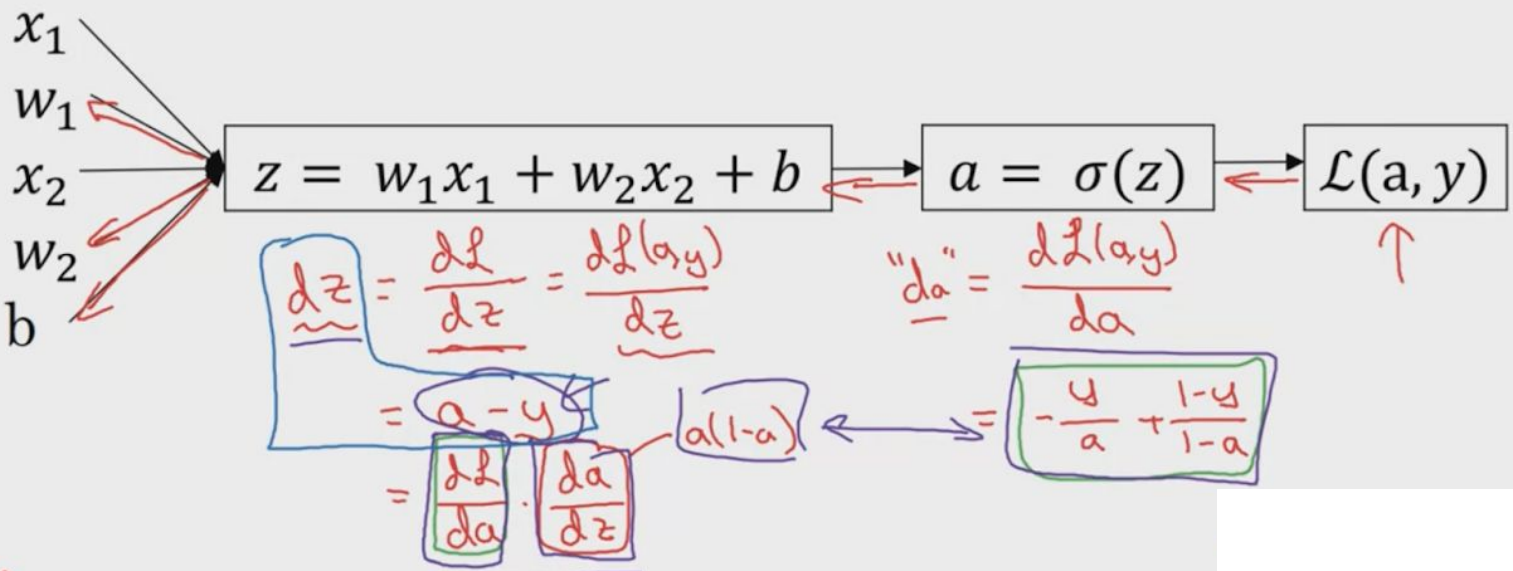


Ahora hablemos de cómo puede ir hacia atrás para calcular las derivadas

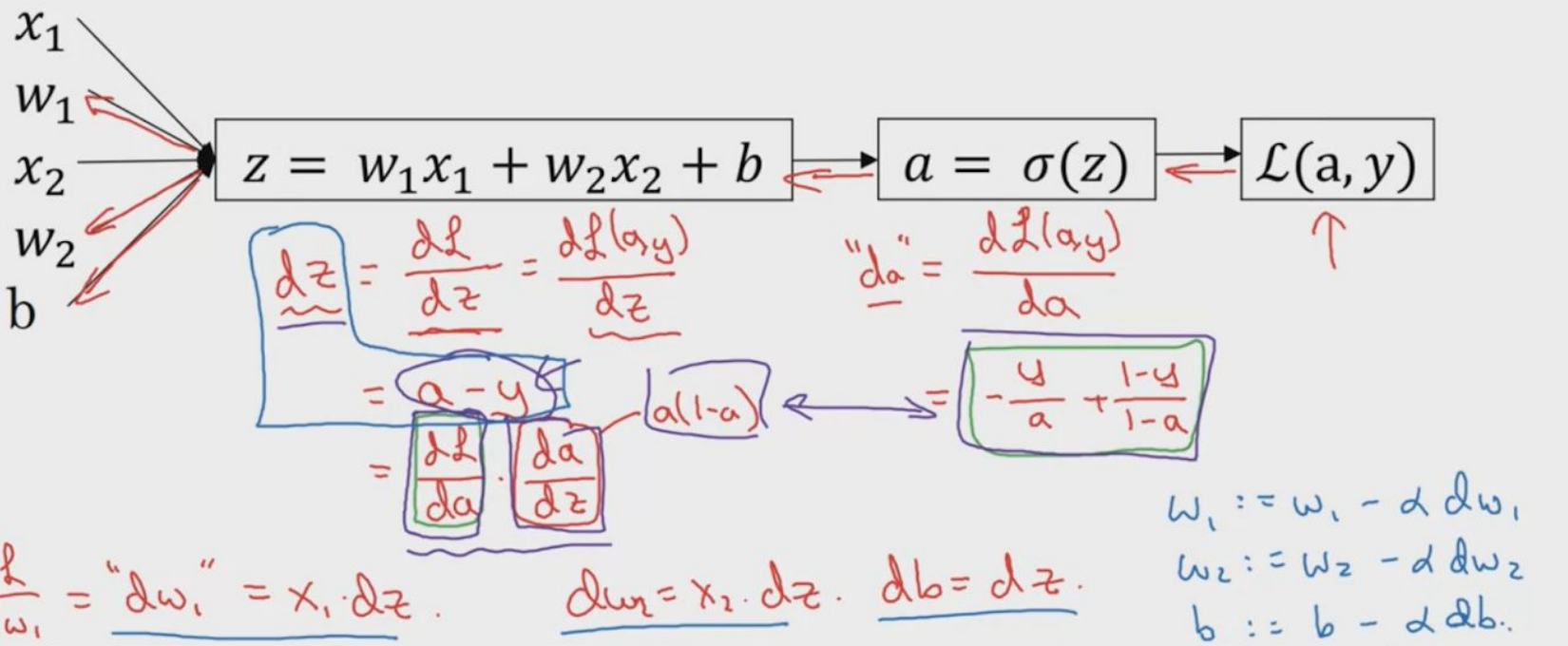
Logistic regression derivatives



Logistic regression derivatives



Logistic regression derivatives



y B se establece como B menos la tasa de aprendizaje por DB.

Andrew Ng

Pregunta

In this video, what is the simplified formula for the derivative of the losswith respect to z?

- a - y
- a (1 - y)
- a / (1-a)

Pregunta

In this video, what is the simplified formula for the derivative of the losswith respect to z?

- [x] a - y
- [ ] a (1 - y)
- [ ] a / (1-a)

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Recap

- Cómo calcular las derivadas, pero con respecto a un solo ejemplo

Let’s see

- Cómo se realiza el cálculo de las derivadas para m ejemplos de entrenamiento

Logistic regression on m examples

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

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

$(x^{(i)}, y^{(i)})$

$dw_1^{(i)}$, $dw_2^{(i)}$, $db^{(i)}$

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

$dw_1^{(i)}$ – $(x^{(i)}, y^{(i)})$

Implementar regresión logística con pendiente descendente trabajando

Logistic regression on m examples

$J=0$; $dw_1=0$; $dw_2=0$; $db=0$

For $i=1$ to m

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

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

$J \mathrel{+}= -\left[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)})\right]$

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

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

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

dw_3 ... dw_n

$db \mathrel{+}= dz^{(i)}$

$n=2$

$J /= m$

$dw_1 /= m$; $dw_2 /= m$; $db /= m$.

$$dw_1 = \frac{\partial J}{\partial w_1}$$

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

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

$b := b - \alpha\, db$.

Vectorization

implementa todo esto sin usar siquiera un solo bucle for

Pregunta

In the for loop depicted in the video, why is there only one *dw* variable (i.e. no i superscripts in the for loop)?

- ○ The value of *dw* in the code is cumulative.
- ○ Only one derivative is being computed.
- ○ Only the derivative of one value is relevant.

Pregunta

In the for loop depicted in the video, why is there only one *dw* variable (i.e. no i superscripts in the for loop)?

- (●) The value of *dw* in the code is cumulative.
- ( ) Only one derivative is being computed.
- ( ) Only the derivative of one value is relevant.

Next class

- Aplicar el algoritmo de GD vectorizado para evitar lo más que se pueda el uso de ciclos

Resources

Implementación en python

- https://github.com/perborgen/LogisticRegression/blob/master/logistic.py
- https://github.com/SSaishruthi/LogisticRegression_Vectorized_Implementation/blob/master/Logistic_Regression.ipynb