Cross Entropy vs NLLLoss

Index

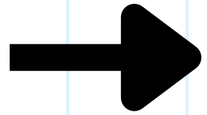
The basics

- Permite evaluar qué tan buenos son los parámetros actuales.
- Este indicador nos permite realizar ajustes a nuestros parámetros/pesos tratando de minimizarla .

Categorical Cross Entropy

Categorical Cross Entropy

Binary Cross Entropy

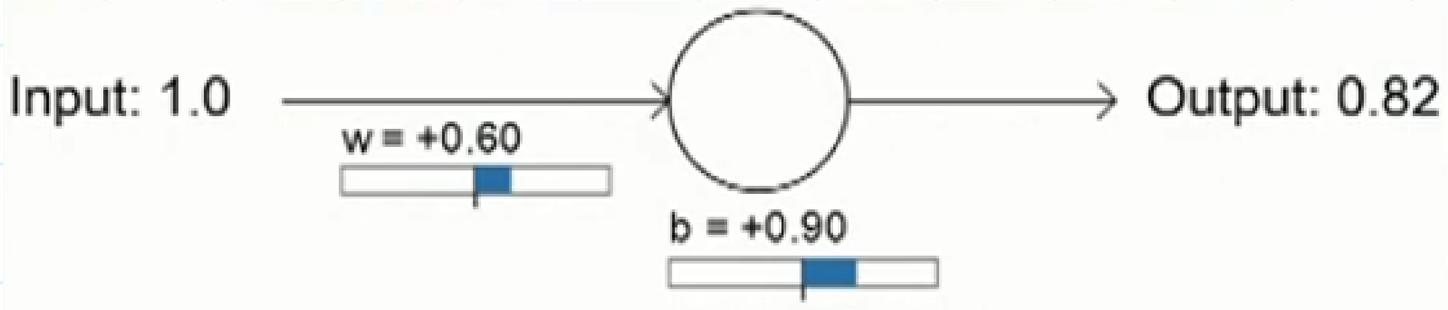


Categorical Cross Entropy

Binary Cross Entropy → Categorical Cross Entropy

¿Por qué usar BCELoss y no otra?

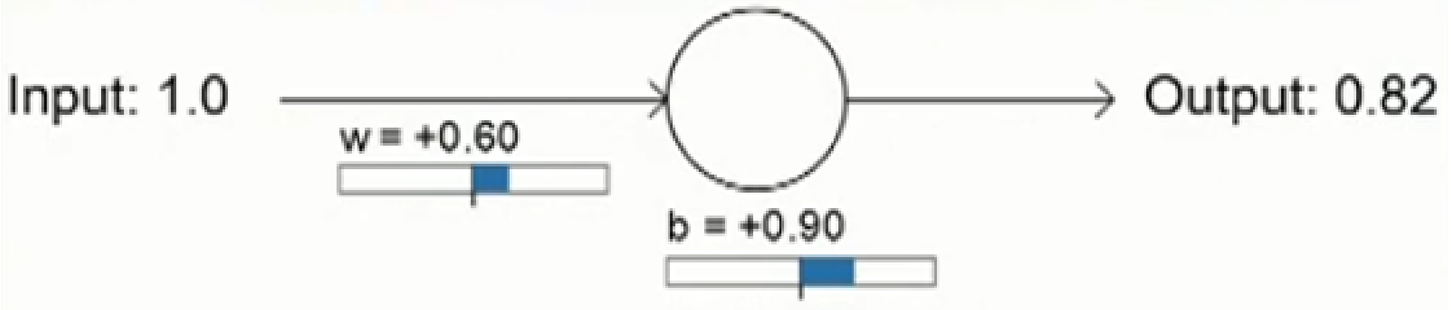
¿Qué pasa si usamos MSE?



$$y = Desired\ Output = 0.0$$
$$x = Input = 1.0$$
$$b_{start} = 0.9$$
$$w_{start} = 0.6$$
$$z = wx + b$$
$$a = \sigma(z)$$

$$C = \frac{(y-a)^2}{2}$$

Binary Cross-Entropy



$$y = Desired\ Output = 0.0$$
$$x = Input = 1.0$$
$$b_{start} = 0.9$$
$$w_{start} = 0.6$$
$$z = wx + b$$
$$a = \sigma(z)$$

$$C = \frac{(y-a)^2}{2}$$

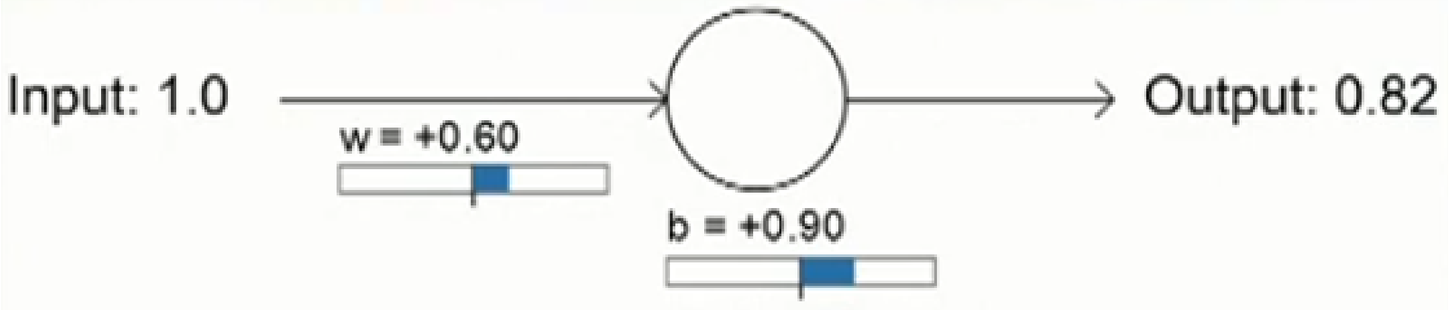
Primer epoch

$$z = 1(0.6) + 0.9 = 1.5$$

$$a = \frac{1}{1+e^{-1.5}} = 0.82$$

$$C = \frac{(0-0.82)^2}{2} = 0.0162$$

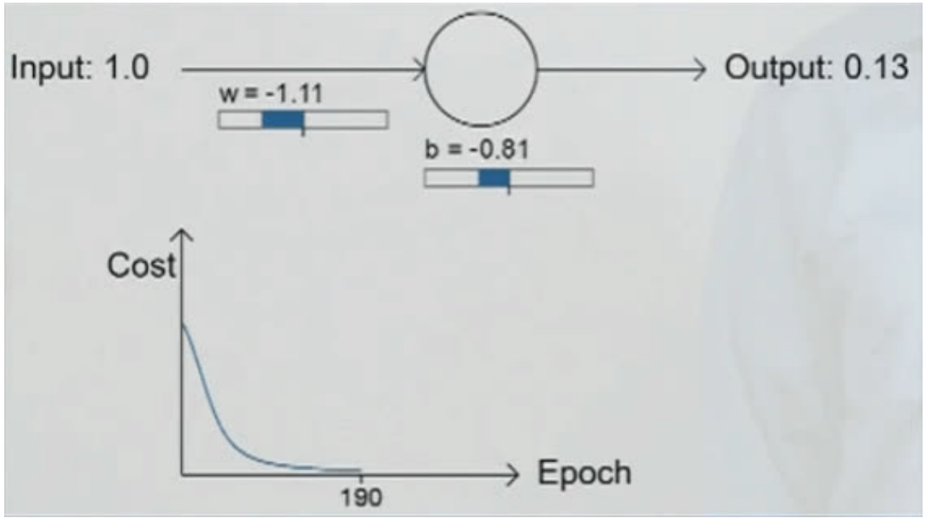
Binary Cross-Entropy



$$y = Desired\ Output = 0.0$$
$$x = Input = 1.0$$
$$b_{start} = 0.9$$
$$w_{start} = 0.6$$
$$z = wx + b$$
$$a = \sigma(z)$$

$$C = \frac{(y-a)^2}{2}$$

300 epochs



Binary Cross-Entropy

¿Y si empezamos con pesos malos?

$$y = Desired\ Output = 0.0$$
$$x = Input = 1.0$$
$$b_{start} = 2.0$$
$$w_{start} = 2.0$$
$$z = wx + b$$
$$a = \sigma(z)$$

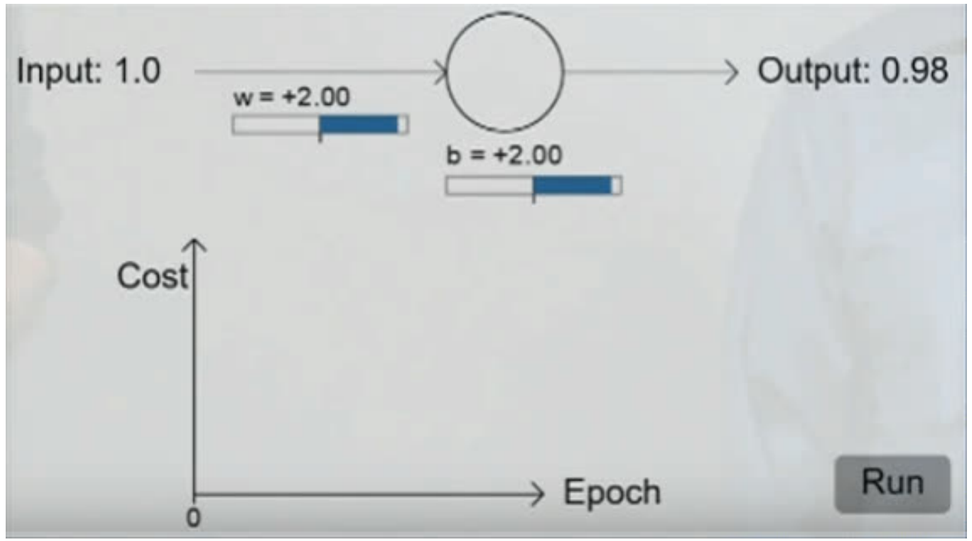
$$C = \frac{(y-a)^2}{2}$$

Binary Cross-Entropy

¿Y si empezamos con pesos malos?

$$y = Desired\ Output = 0.0$$
$$x = Input = 1.0$$
$$b_{start} = 2.0$$
$$w_{start} = 2.0$$
$$z = wx + b$$
$$a = \sigma(z)$$
$$C = \frac{(y-a)^2}{2}$$

300 epochs

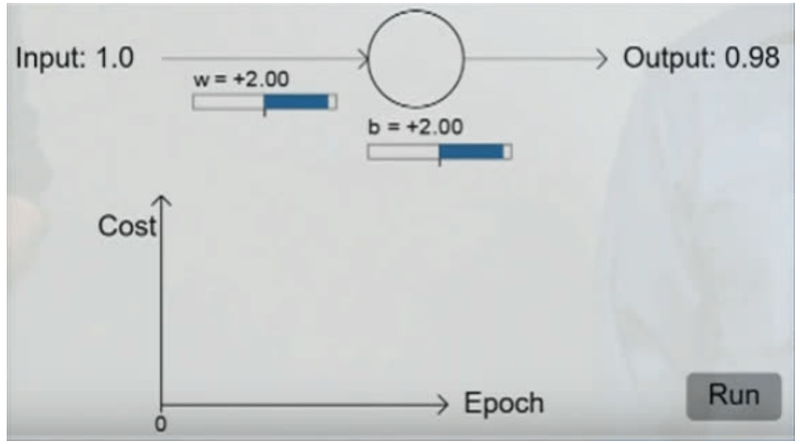


Binary Cross-Entropy

¿Por qué pasa esto?

$$y = Desired\ Output = 0.0$$
$$x = Input = 1.0$$
$$b_{start} = 2.0$$
$$w_{start} = 2.0$$
$$z = wx + b$$
$$a = \sigma(z)$$
$$C = \frac{(y-a)^2}{2}$$

300 epochs



Binary Cross-Entropy

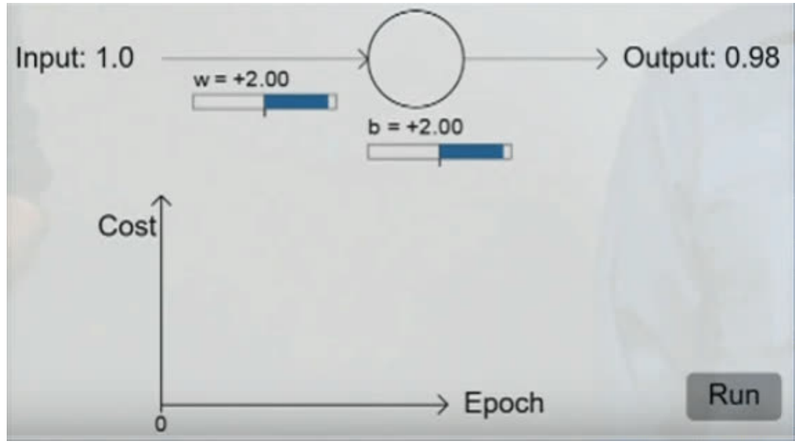
¿Por qué pasa esto?

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

Encontremos el gradiente

300 epochs



Binary Cross-Entropy

¿Por qué pasa esto?

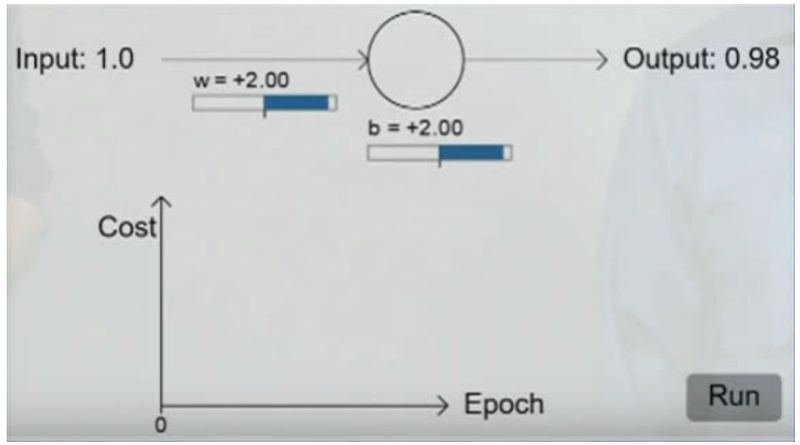
$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

Encontremos el gradiente

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

300 epochs



Binary Cross-Entropy

¿Por qué pasa esto?

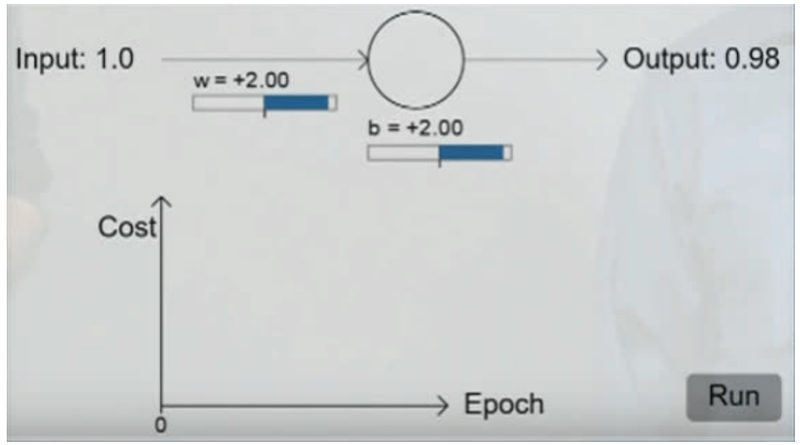
$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

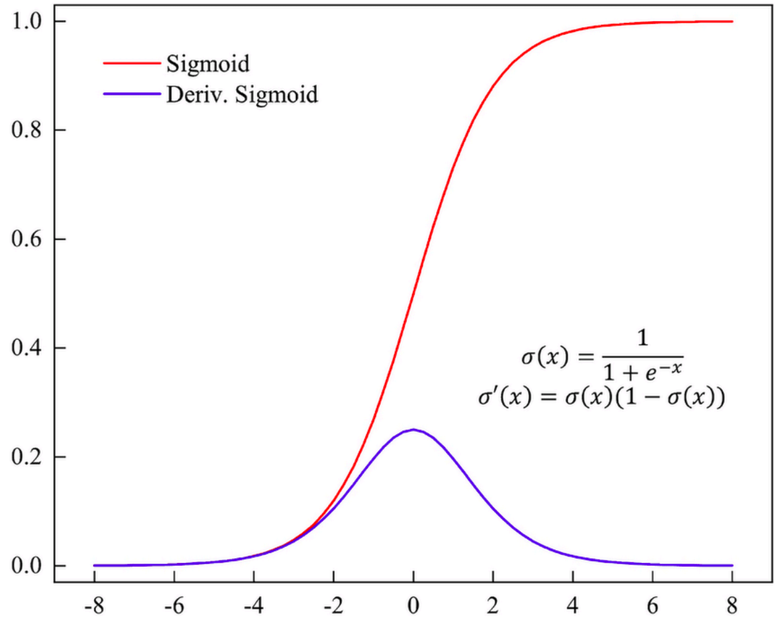
$$C = \frac{(y-a)^2}{2}$$

Encontremos el gradiente

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

300 epochs





Binary Cross-Entropy

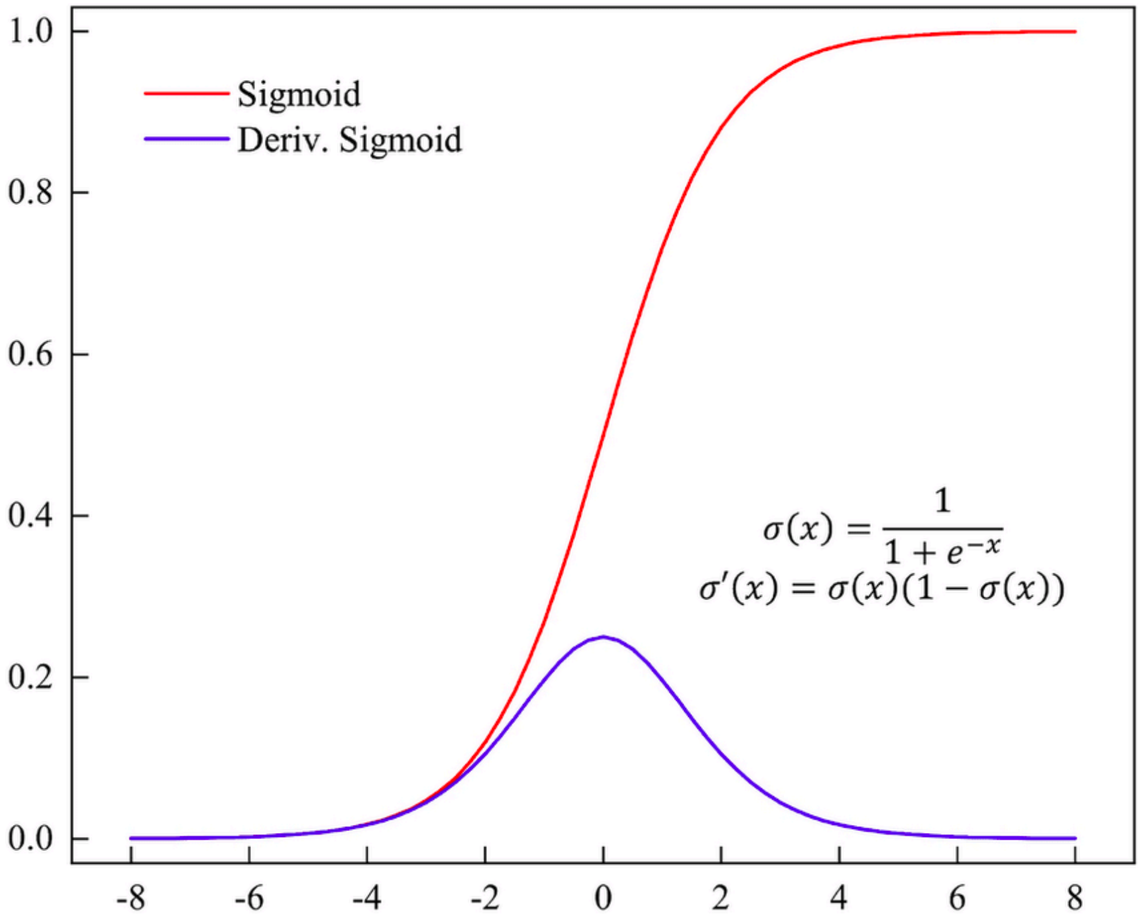
¿Por qué pasa esto?

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

Encontremos el gradiente

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

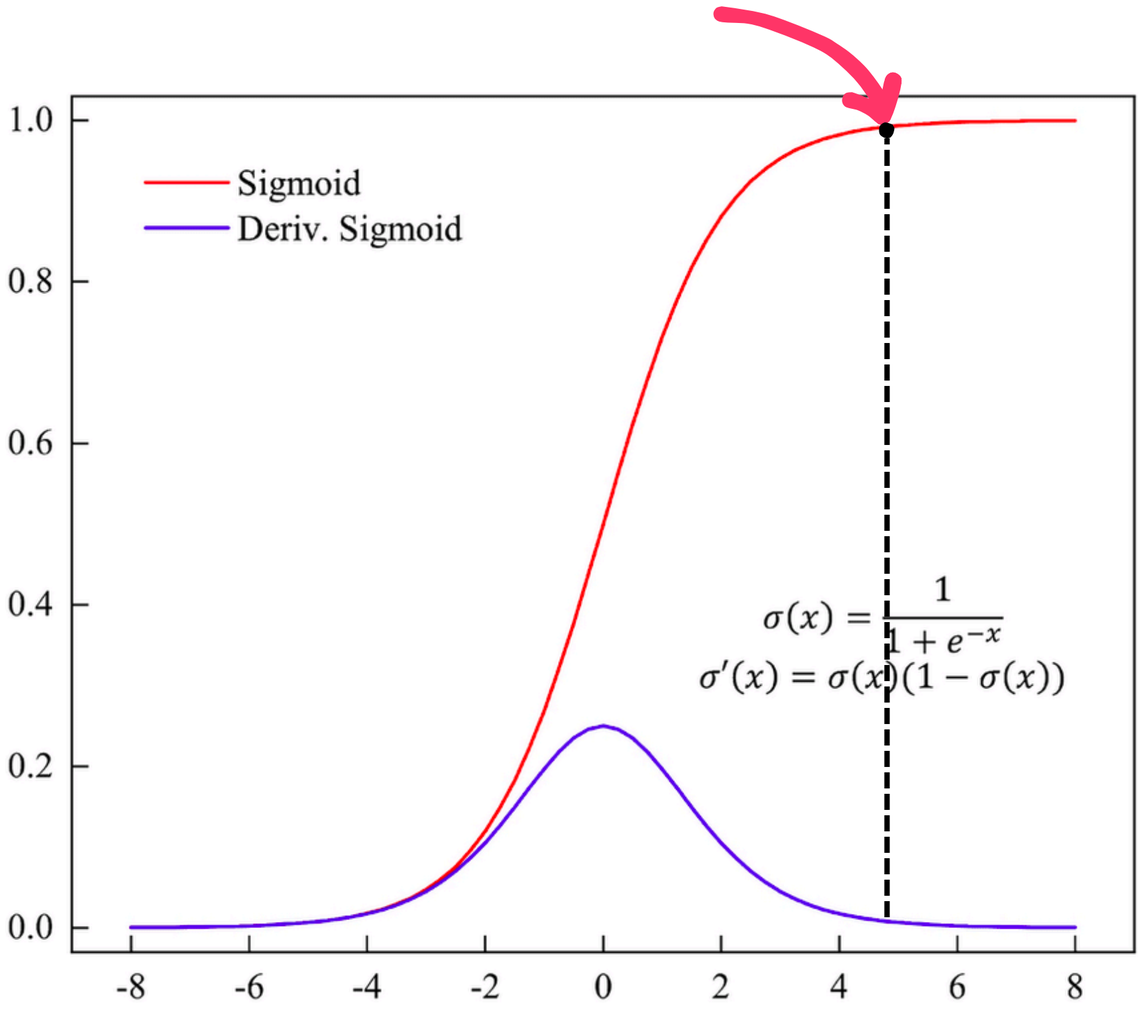


Binary Cross-Entropy

¿Por qué pasa esto?

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$



Encontremos el gradiente

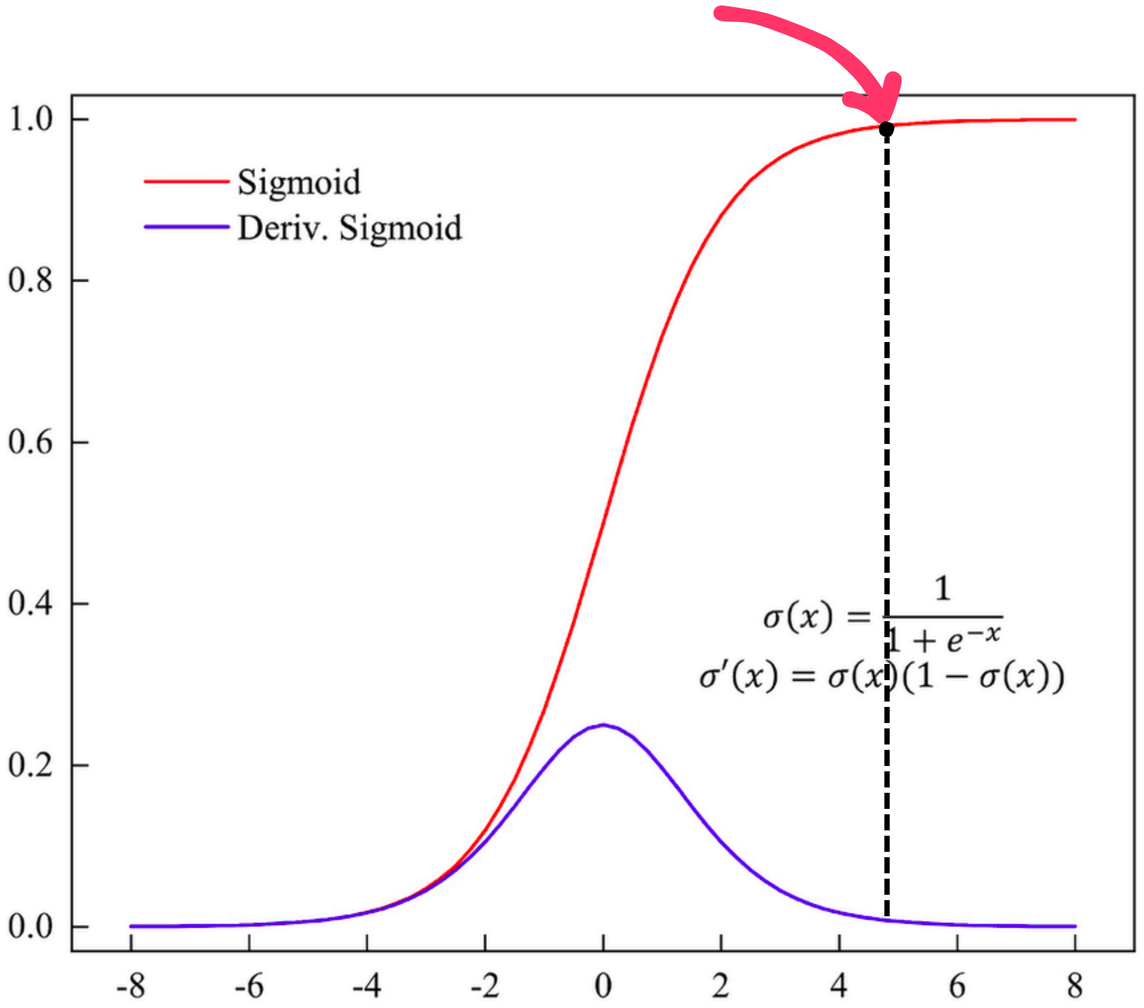
$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

Binary Cross-Entropy

¿Por qué pasa esto?

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$



Encontremos el gradiente

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

Binary Cross-Entropy

Binary Cross Entropy

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

Binary Cross-Entropy

Binary Cross Entropy

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

¿Cómo funciona?

Binary Cross-Entropy

Binary Cross Entropy

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

¿Cómo funciona?

Caso y = 0, a = 0.001

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

Binary Cross-Entropy

Binary Cross Entropy

$y = \text{Desired Output} = 0.0$
$x = \text{Input} = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

¿Cómo funciona?

Caso y = 0, a = 0.001

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

Binary Cross-Entropy

Binary Cross Entropy

$y = \text{Desired Output} = 0.0$
$x = \text{Input} = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

¿Cómo funciona?

Caso y = 0, a = 0.001

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

$C \approx 0$

Binary Cross-Entropy

Binary Cross Entropy

$y = \text{Desired Output} = 0.0$
$x = \text{Input} = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

¿Cómo funciona?

Caso y = 0, a = 0.001

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

$C \approx 0$

Caso y = 1, a = 0.999

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

Binary Cross-Entropy

Binary Cross Entropy

$y = \text{Desired Output} = 0.0$
$x = \text{Input} = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

¿Cómo funciona?

Caso y = 0, a = 0.001

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

$C \approx 0$

Caso y = 1, a = 0.999

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

Binary Cross-Entropy

Binary Cross Entropy

$y = \text{Desired Output} = 0.0$
$x = \text{Input} = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

¿Cómo funciona?

Caso y = 0, a = 0.001

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

$C \approx 0$

Caso y = 1, a = 0.999

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

$C \simeq 0$

Binary Cross-Entropy

Binary Cross Entropy

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

¿Y el gradiente?

Binary Cross-Entropy

Binary Cross Entropy

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x \left[y \ln a + (1-y)\ln(1-a)\right],$$

¿Y el gradiente?

$$\frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j}$$
$$= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j.$$

Binary Cross-Entropy

Binary Cross Entropy

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x \left[y \ln a + (1-y)\ln(1-a)\right],$$

¿Y el gradiente?

$$\frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j}$$
$$= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j.$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x \frac{\sigma'(z)x_j}{\sigma(z)(1-\sigma(z))}(\sigma(z)-y).$$

Binary Cross-Entropy

Binary Cross Entropy

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x \left[y \ln a + (1-y)\ln(1-a)\right],$$

¿Y el gradiente?

$$\frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j}$$
$$= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j.$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x \frac{\sigma'(z)x_j}{\sigma(z)(1-\sigma(z))}(\sigma(z)-y).$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z)-y).$$

Binary Cross-Entropy

Binary Cross Entropy

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$C = -\frac{1}{n}\sum_x \left[y \ln a + (1-y)\ln(1-a)\right],$$

¿Y el gradiente?

$$\frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j}$$
$$= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j.$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x \frac{\sigma'(z)x_j}{\sigma(z)(1-\sigma(z))}(\sigma(z)-y).$$

Sin $\sigma'(z)$ ✓

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z)-y).$$

Binary Cross-Entropy

Binary Cross Entropy en nuestro modelo

Pesos ‘buenos’

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 0.9$
$w_{start} = 0.6$
$z = wx + b$
$a = \sigma(z)$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

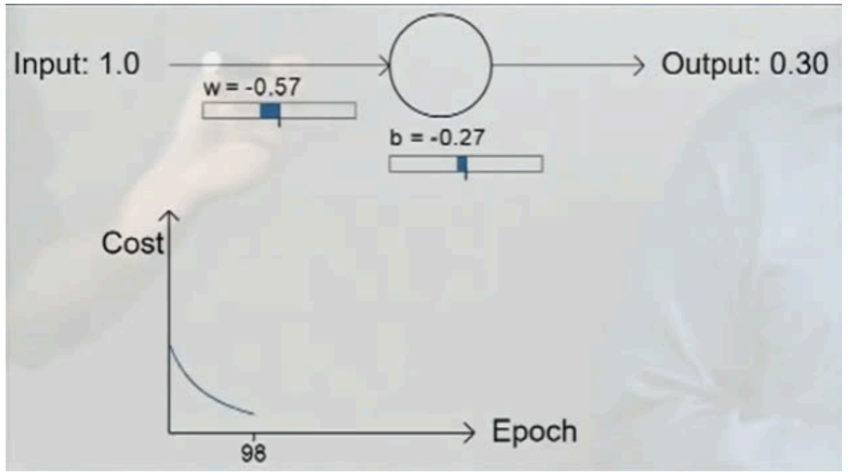
Binary Cross-Entropy

Binary Cross Entropy en nuestro modelo

Pesos ‘buenos’

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 0.9$
$w_{start} = 0.6$
$z = wx + b$
$a = \sigma(z)$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$



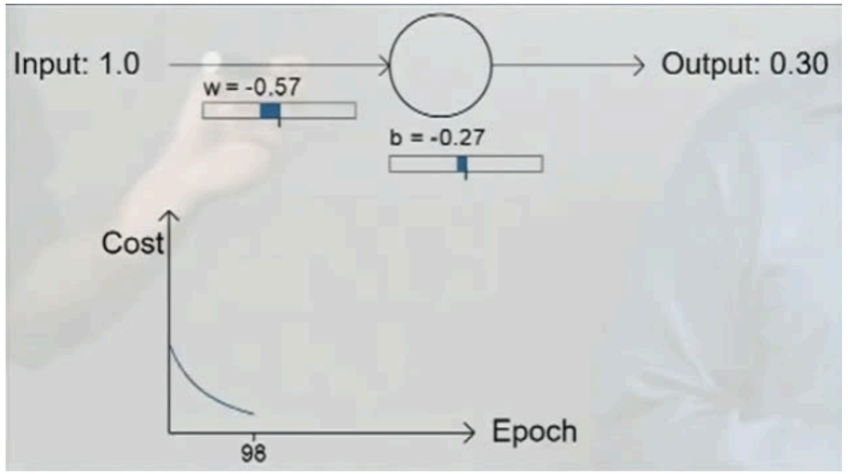
Binary Cross-Entropy

Binary Cross Entropy en nuestro modelo

Pesos ‘buenos’

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 0.9$
$w_{start} = 0.6$
$z = wx + b$
$a = \sigma(z)$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$



Pesos ‘malos’

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

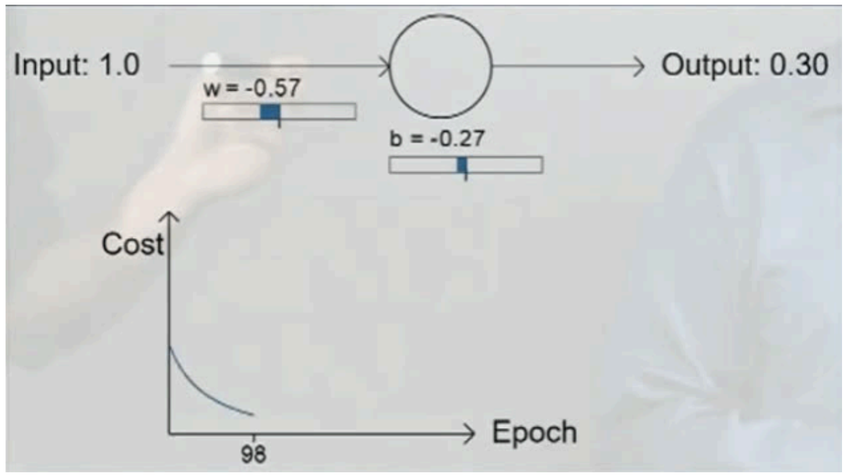
Binary Cross-Entropy

Binary Cross Entropy en nuestro modelo

Pesos ‘buenos’

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 0.9$
$w_{start} = 0.6$
$z = wx + b$
$a = \sigma(z)$

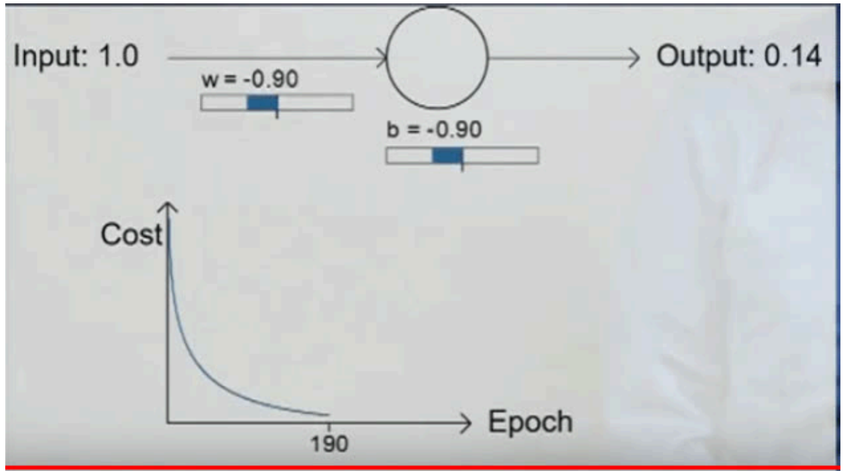
$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$



Pesos ‘malos’

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$



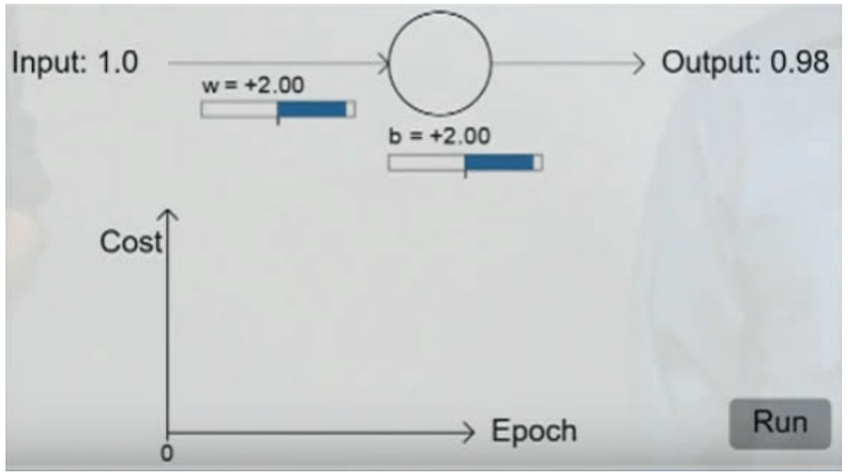
Binary Cross-Entropy

Binary Cross Entropy en nuestro modelo

Con MSE

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

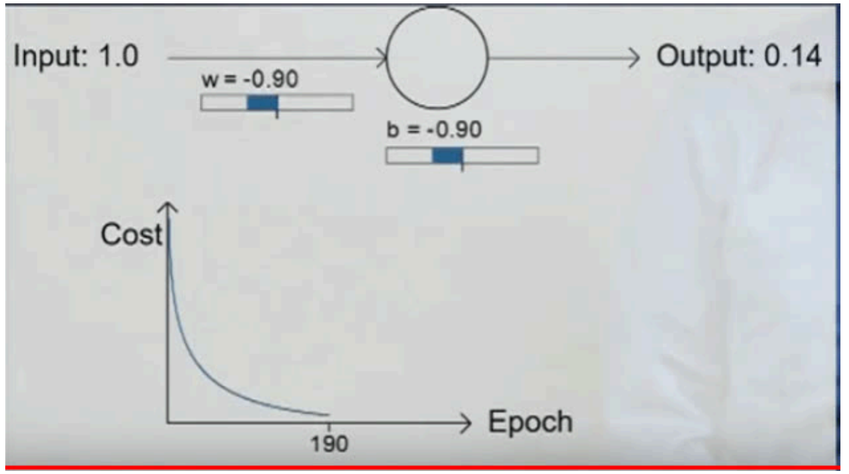
$$C = \frac{(y-a)^2}{2}$$



Con BCELoss

$y = Desired\ Output = 0.0$
$x = Input = 1.0$
$b_{start} = 2.0$
$w_{start} = 2.0$
$z = wx + b$
$a = \sigma(z)$

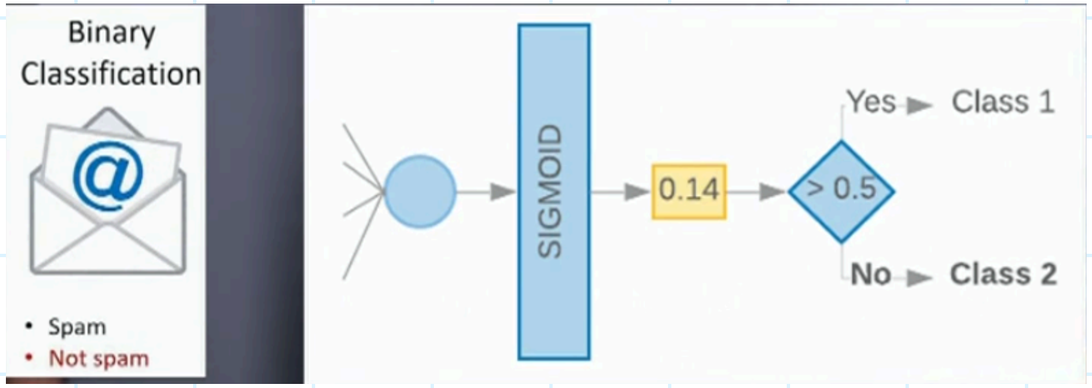
$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$



Binary Cross-Entropy

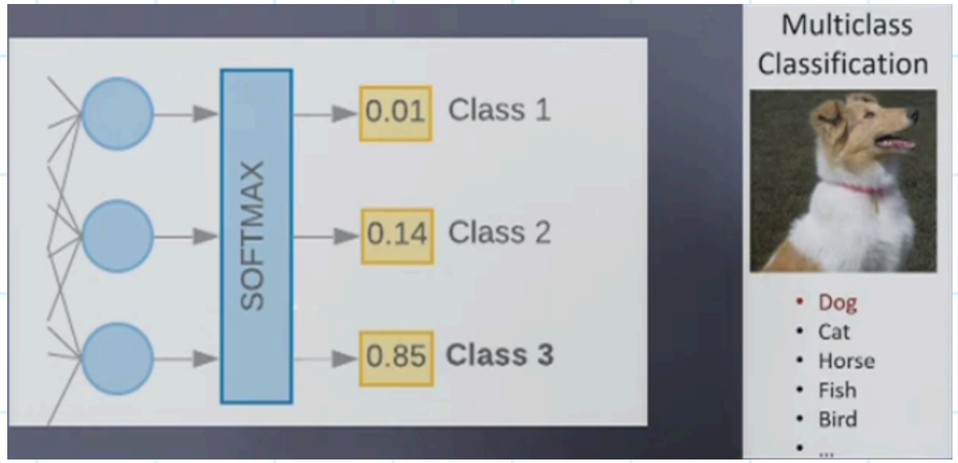
Binary Cross Entropy

$$C = -\frac{1}{n}\sum_x \left[y \ln a + (1-y)\ln(1-a)\right],$$



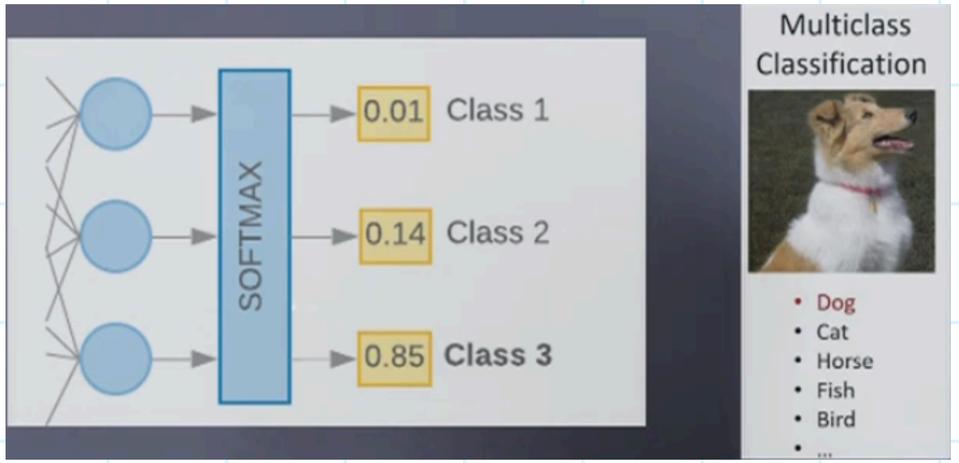
Categorical Cross Entropy

$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$



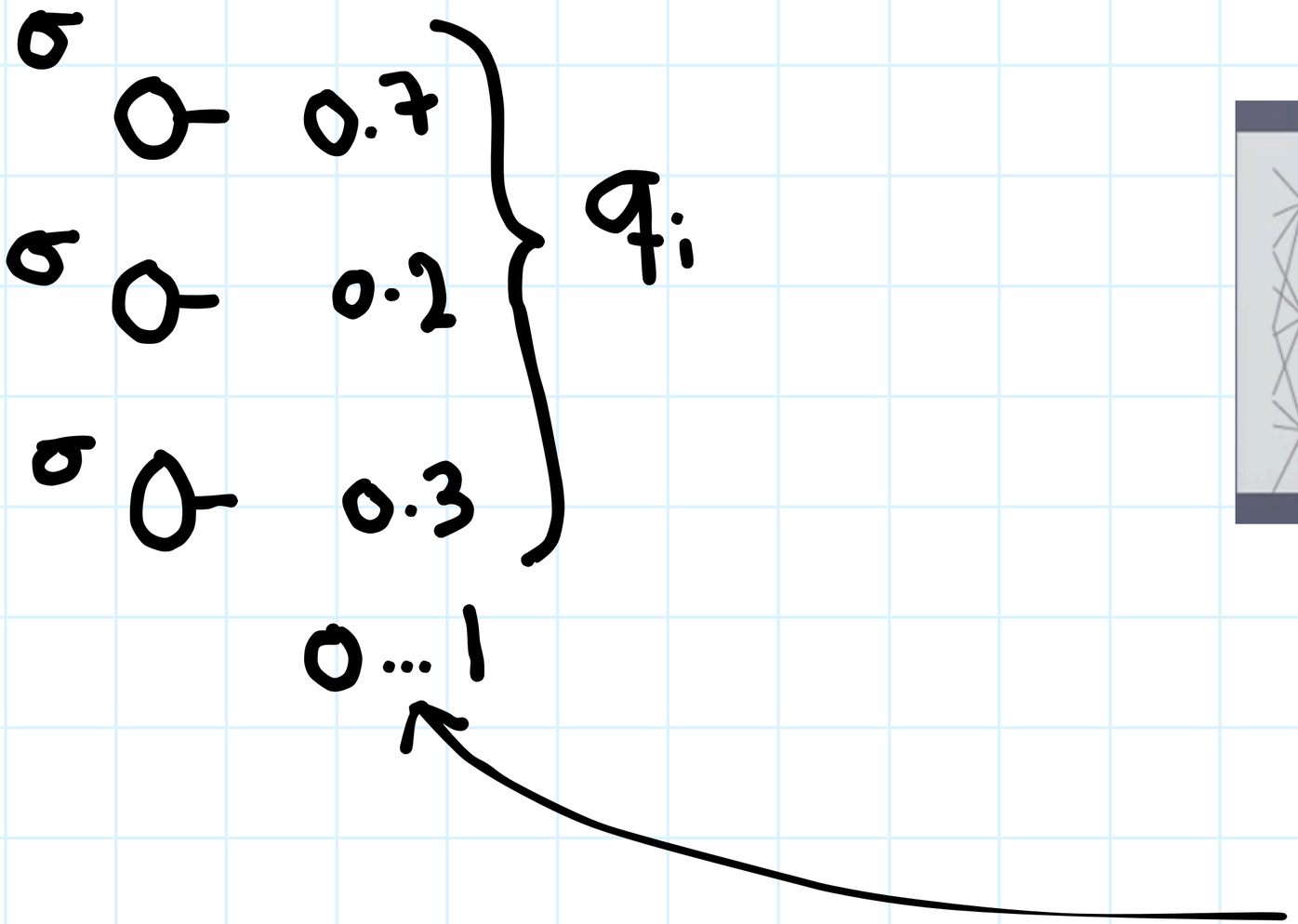
¿Por qué softmax?

$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$

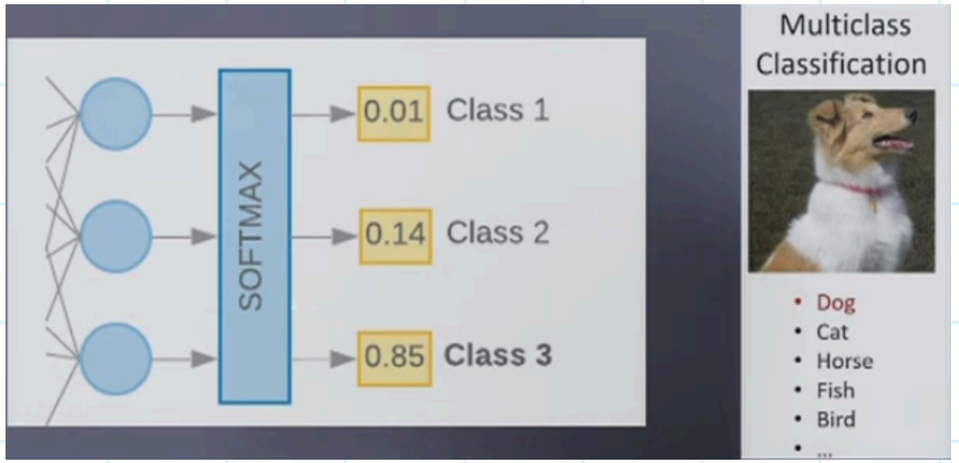


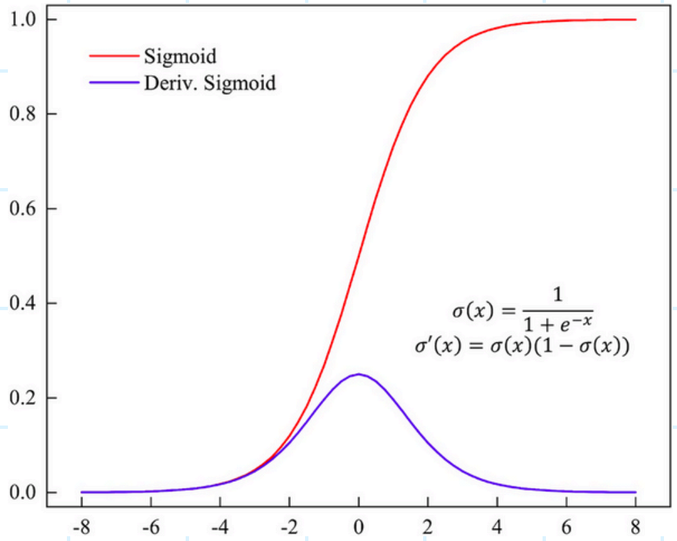
Categorical Cross - Entropy Loss Softmax

¿Por qué softmax?



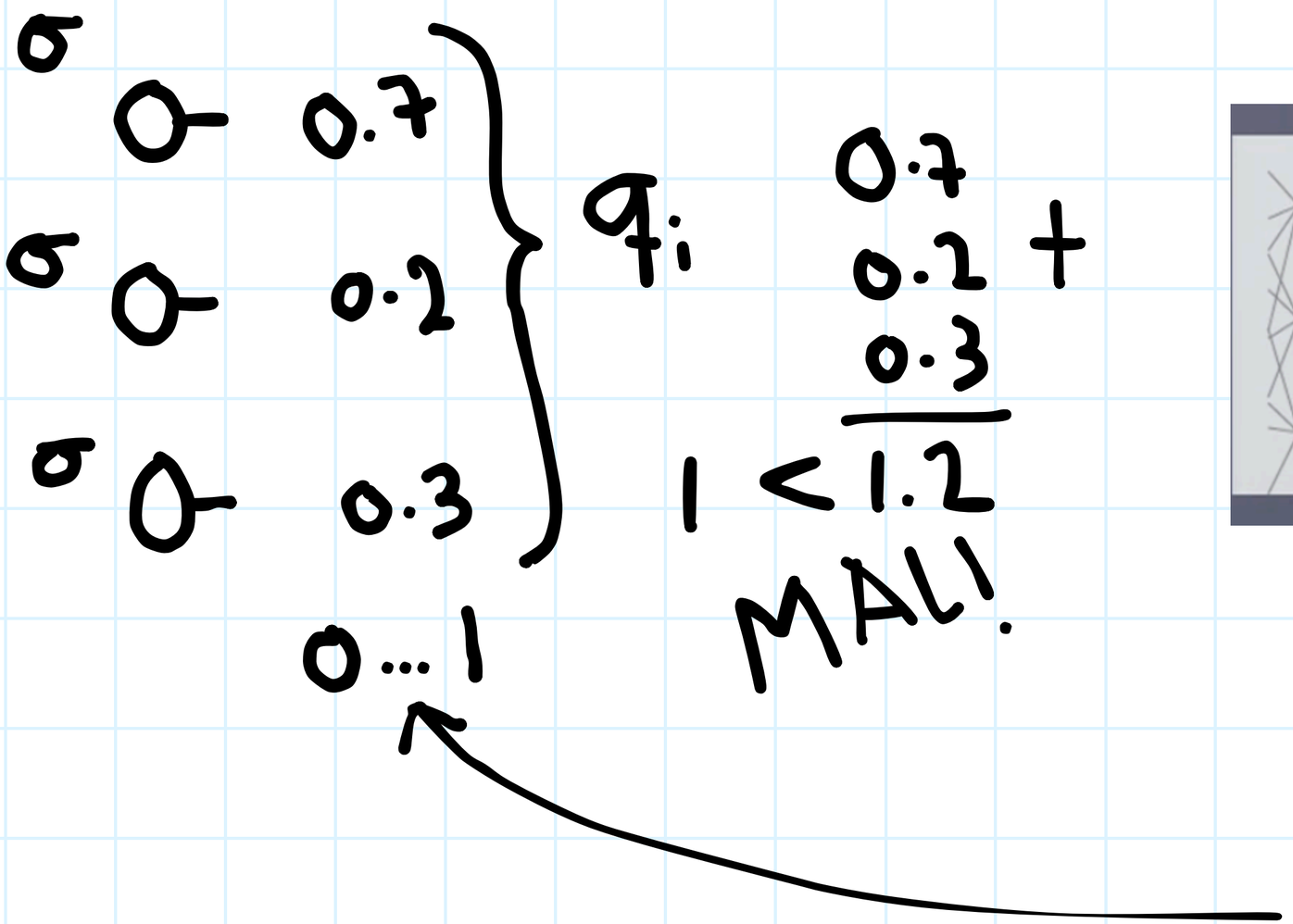
$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$



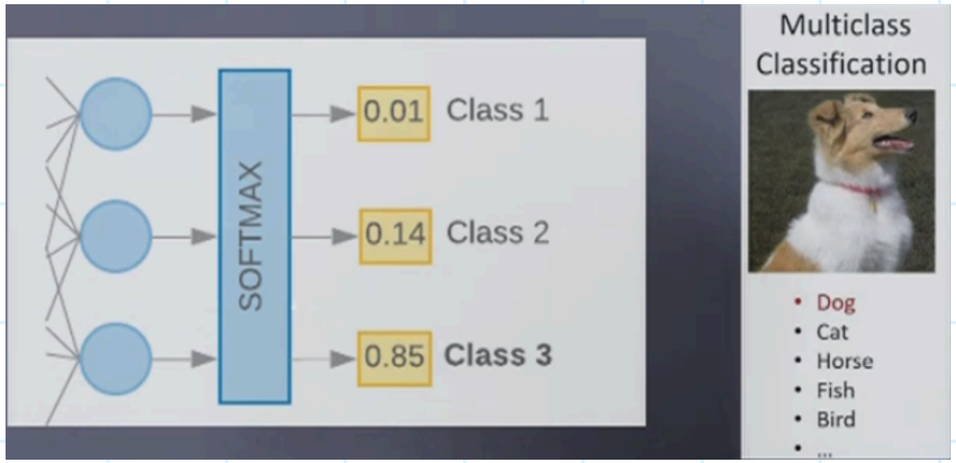


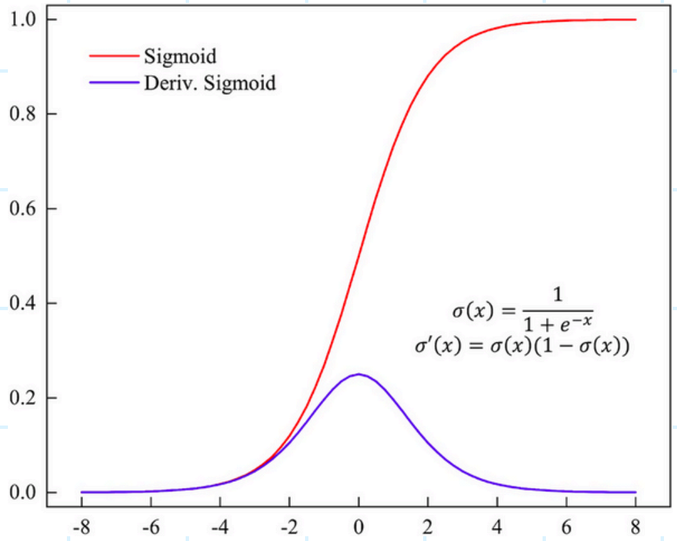
Categorical Cross - Entropy Loss Softmax

¿Por qué softmax?



$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$

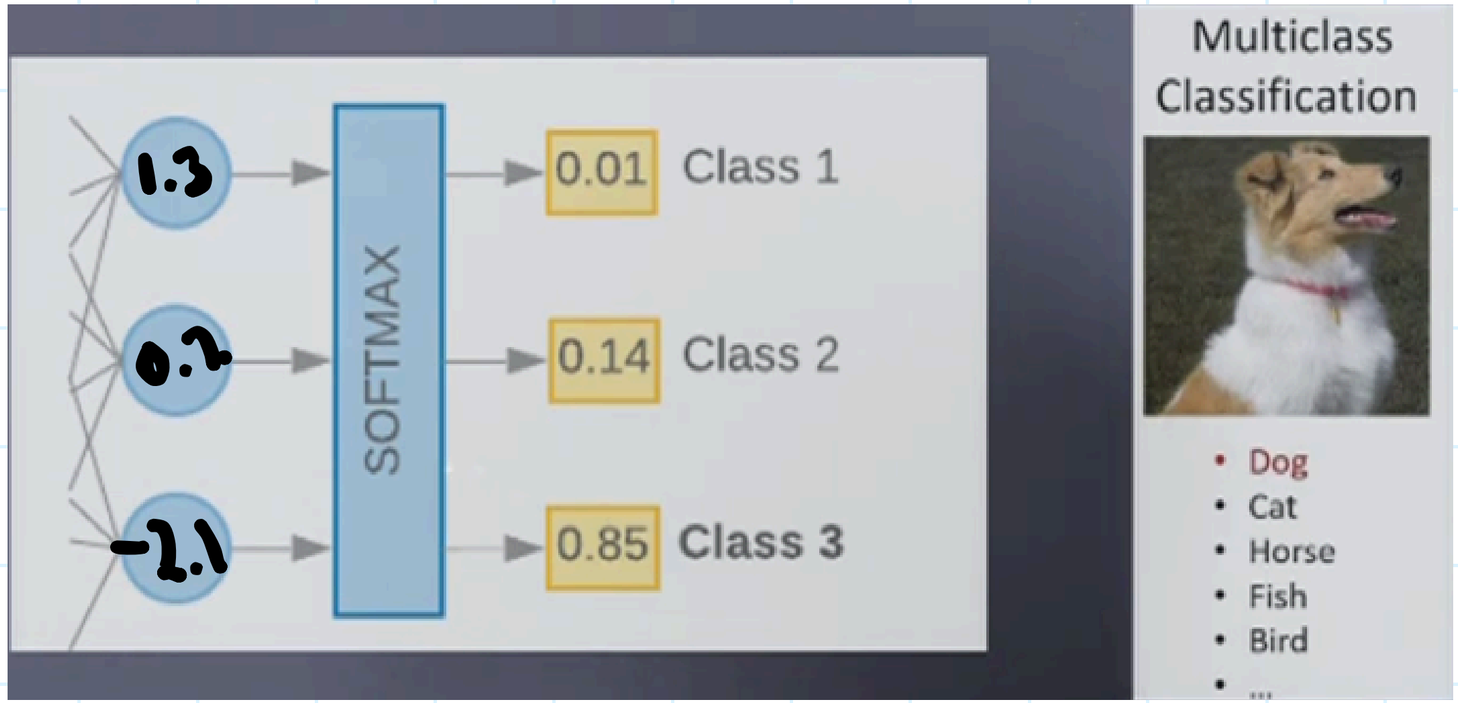




Categorical Cross - Entropy Loss Softmax

¿Por qué softmax?

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \; for\; j = 1, \ldots, k$$



$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$

$$e^{1.3} + e^{0.2} + e^{-2.1} = 5.013155..$$

$$\frac{e^{1.3}}{5.013155} = 0.731..$$

$$\frac{e^{0.2}}{5.013155} = 0.243..$$

$$\frac{e^{-2.1}}{5.013155} = 0.024..$$

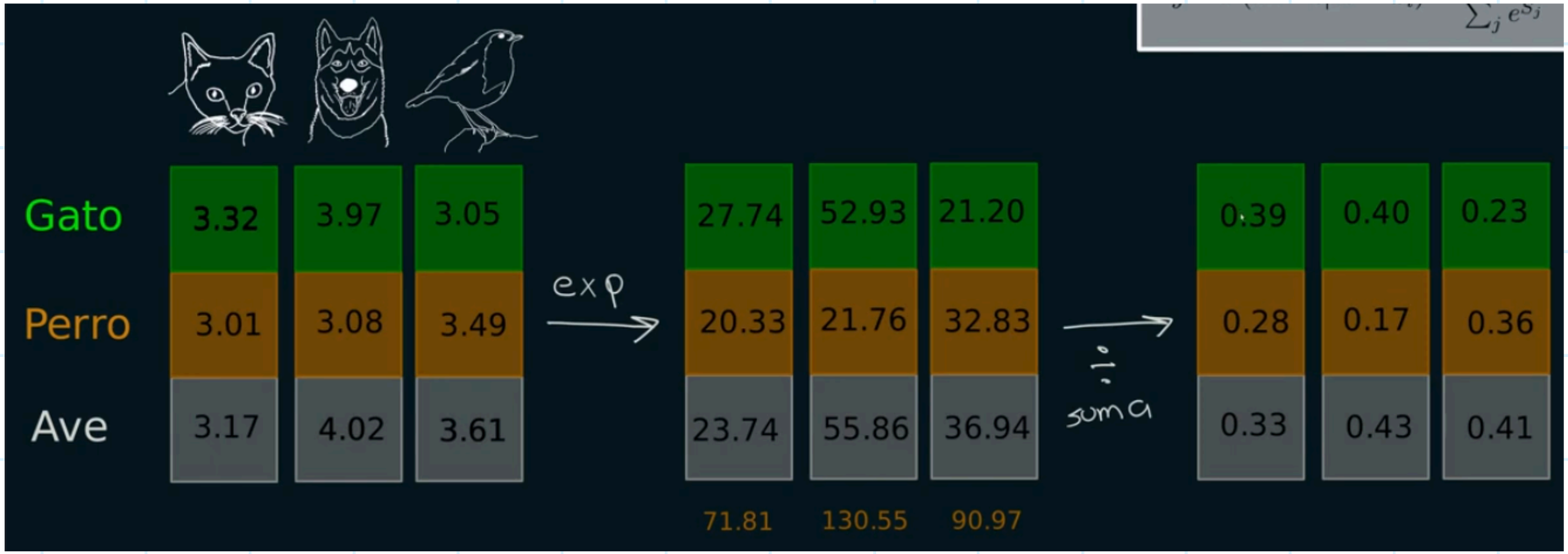
$$0.024 + 0.243 + 0.731 = 1$$

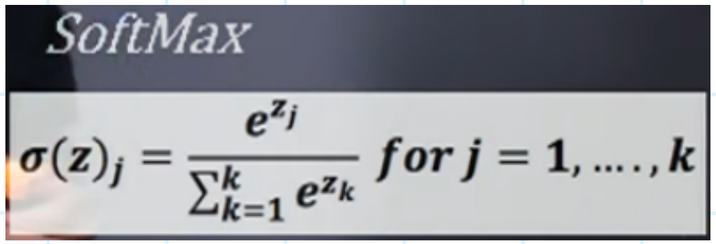
Categorical Cross - Entropy Loss Softmax

Ejemplo

$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$

Batch size = 3, # Classes = 3



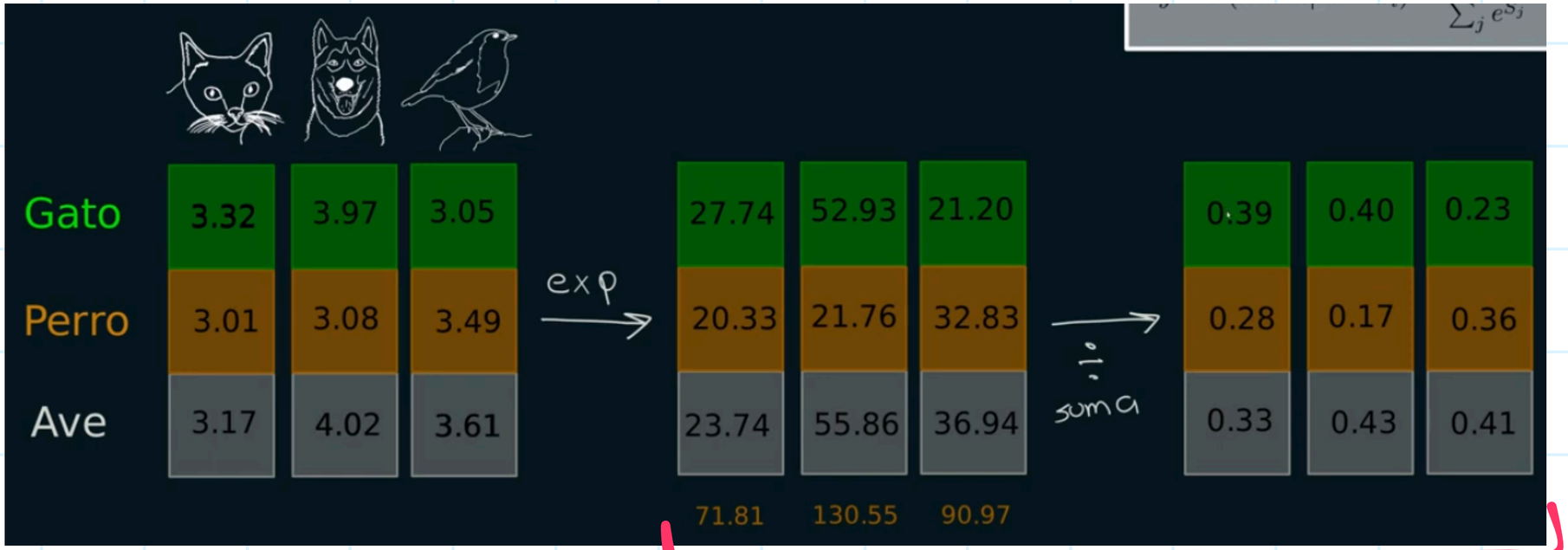


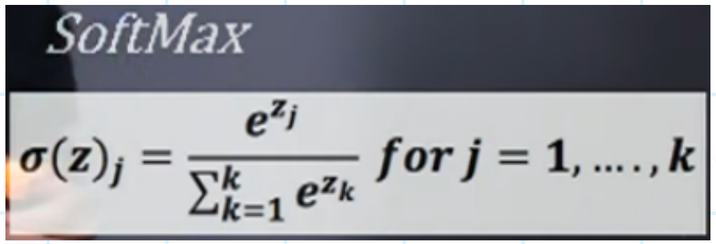
Función de Costo - Cross Entropy

Ejemplo

$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$

Batch size = 3, # Classes = 3





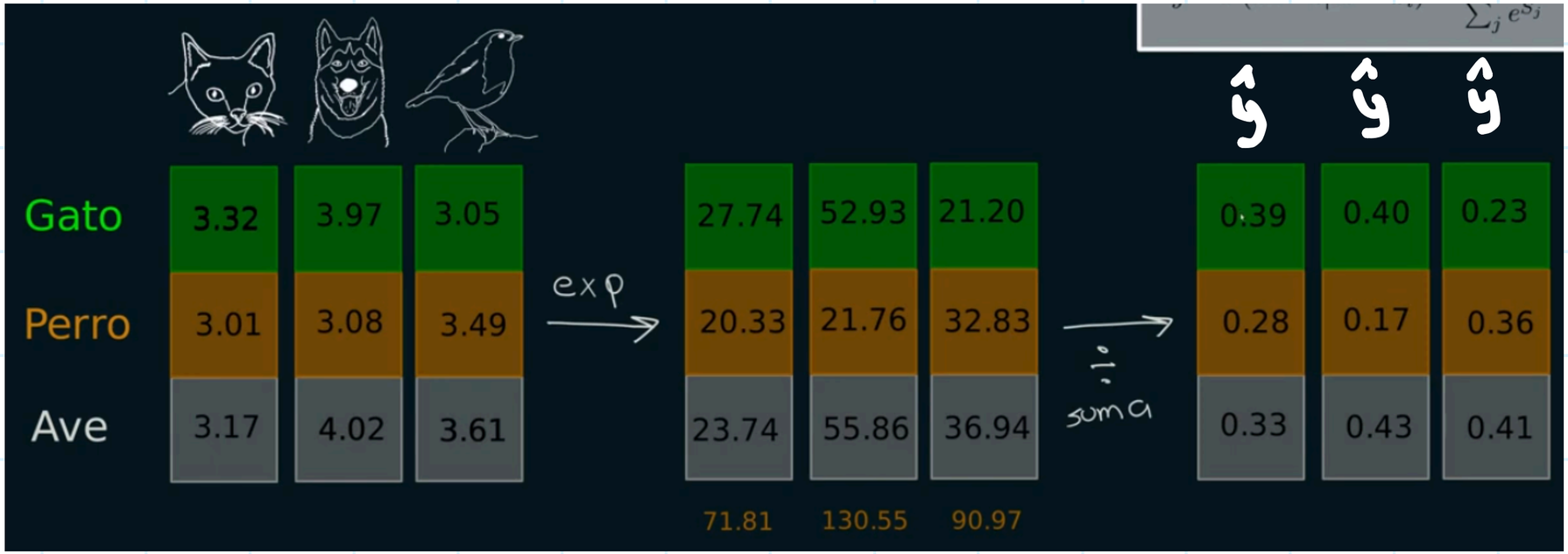
Softmax

Función de Costo - Cross Entropy

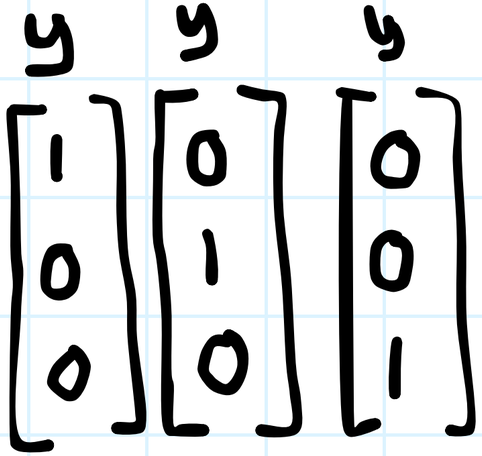
Ejemplo

$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i) \rightarrow L_i = -\sum_{j=1}^{\#clases} y_j ln(\hat{y}_j) = -ln\left(\frac{e^{S_k}}{\sum_j e^{S_j}}\right)$$

Batch size = 3, # Classes = 3



One-hot



SoftMax

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \quad for\ j = 1, \ldots, k$$

$$gato\_loss = -(1ln(0.39) + 0ln(0.28) + 0ln(0.33)) = 0.9416$$

$$perro\_loss = -(1ln(0.17)) = 1.7719$$

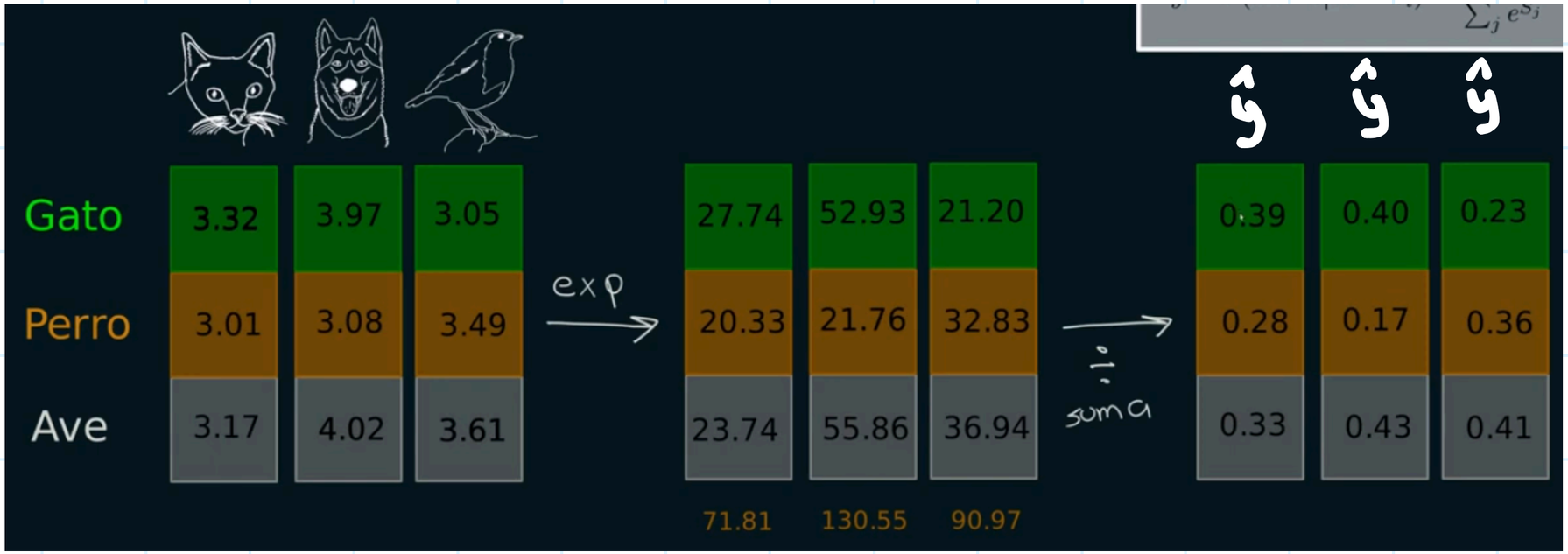
$$ave\_loss = -ln(0.41) = 0.8915$$

Función de Costo - Cross Entropy

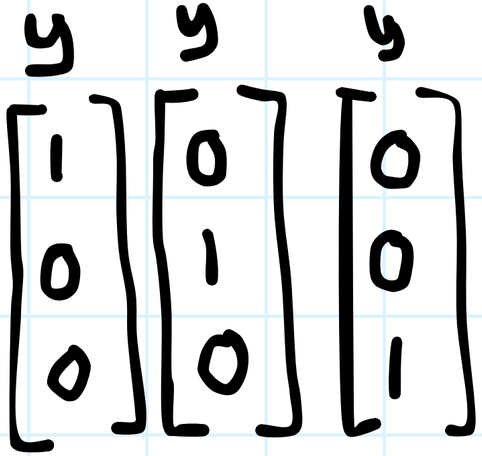
Ejemplo

$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i) \rightarrow L_i = -\sum_{j=1}^{\#clases} y_j ln(\hat{y}_j) = -ln\left(\frac{e^{S_k}}{\sum_j e^{S_j}}\right)$$

Batch size = 3, # Classes = 3



One-hot



SoftMax

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \quad for\ j = 1, \ldots, k$$

$$gato\_loss = -(1ln(0.39) + 0ln(0.28) + 0ln(0.33)) = 0.9416$$

$$perro\_loss = -(1ln(0.17)) = 1.7719$$

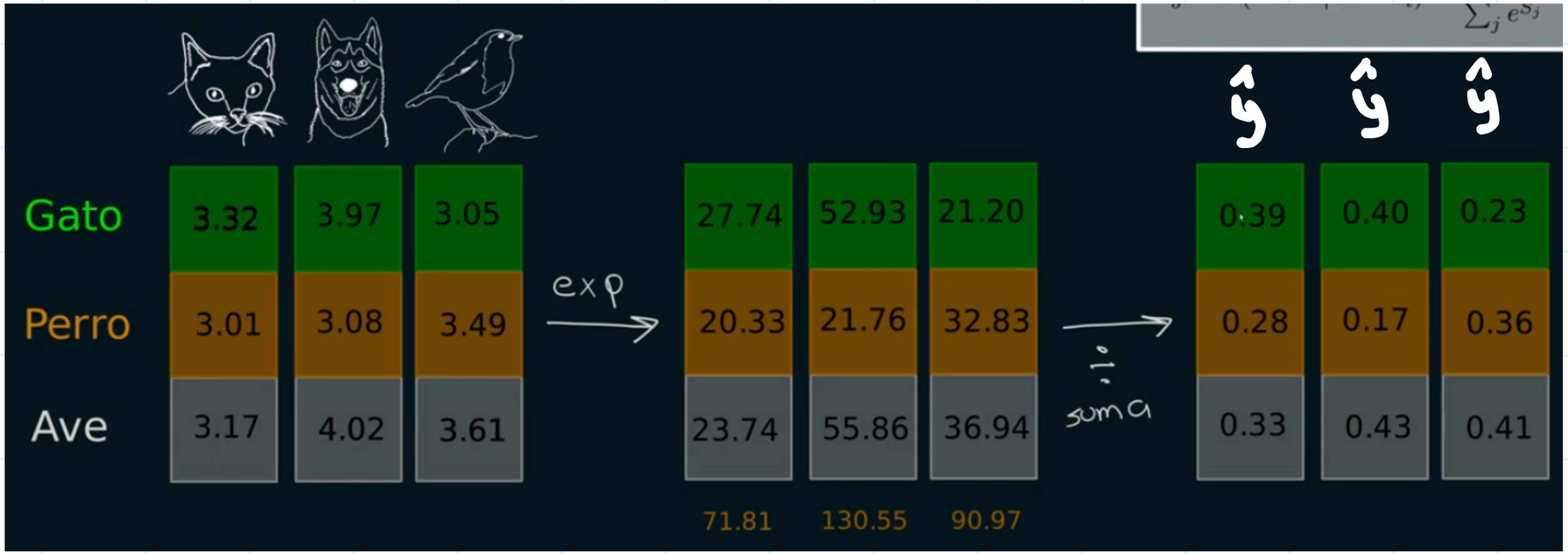
$$ave\_loss = -ln(0.41) = 0.8915 \rightarrow -ln\left(\frac{e^{S_k}}{\sum_j e^{S_j}}\right)$$

Función de Costo - Cross Entropy

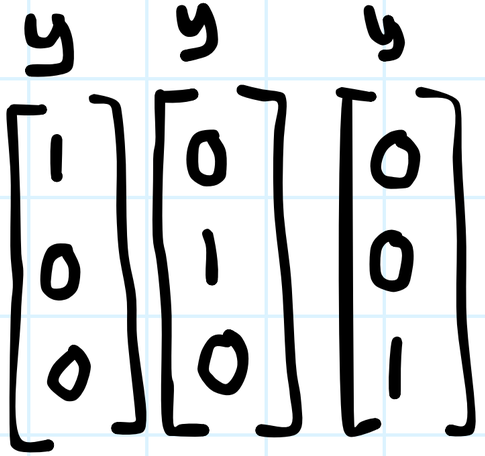
Ejemplo

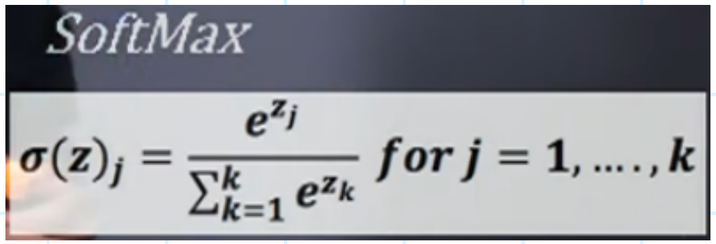
$$-\sum_{i=1}^{\#C} p_i \cdot ln(q_i) \rightarrow L_i = -\sum_{j=1}^{\#clases} y_j ln(\hat{y}_j) = -ln\left(\frac{e^{S_k}}{\sum_j e^{S_j}}\right)$$

Batch size = 3, # Classes = 3



One-hot





$gato\_loss = 0.9416$
$perro\_loss = 1.7719$
$ave\_loss = 0.8915$

¿Loss del batch?

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} -ln\left(\frac{e^{S_k}}{\sum_j e^{S_j}}\right)$$

$$\frac{0.9416+1.7719+0.8915}{3} = 1.2$$

Función de Costo - Cross Entropy

Negative Log Likelihood Loss

Likelihood?

Sacamos 5 pelotas de una bolsa

$p_R = 2/5$

$p_B = 3/5$

Cross-Entropy Loss Log-likelihood Perspective

Likelihood?

q_i = estimated probability of outcome i

p_i = empirical probability of outcome i in the training set

Sacamos 5 pelotas de una bolsa

$$\prod_i q_i^{Np_i} = L$$

$p_R = 2/5$

$p_B = 3/5$

Modelo malo

$q_R = 4/5$

$q_B = 1/5$

$$L = \left(\frac{4}{5}\right)^2 \left(\frac{1}{3}\right)^3 = 0.023$$

Cross-Entropy Loss Log-likelihood Perspective

Likelihood?

q_i = estimated probability of outcome i

p_i = empirical probability of outcome i in the training set

Sacamos 5 pelotas de una bolsa

$$\prod_i q_i^{Np_i} = L$$

$p_R = 2/5$

$p_B = 3/5$

Modelo malo

$q_R = 4/5$

$q_B = 1/5$

$L = \left(\frac{4}{5}\right)^2 \left(\frac{1}{5}\right)^3 = 0.023$

Modelo bueno

$q_R = \frac{2.5}{5}$

$q_B = \frac{2.5}{5}$

$L = \left(\frac{2.5}{5}\right)^2 \left(\frac{2.5}{5}\right)^3 = 0.031$

Cross-Entropy Loss Log-likelihood Perspective

Likelihood?

q_i = estimated probability of outcome i

p_i = empirical probability of outcome i in the training set

Sacamos 5 pelotas de una bolsa

$$\prod_i q_i^{Np_i} = L$$

$p_R = 2/5$

$p_B = 3/5$

Modelo malo

$q_R = 4/5$

$q_B = 1/5$

$L = \left(\frac{4}{5}\right)^2 \left(\frac{1}{3}\right)^3 = 0.023$

$<$

more likeli

Modelo bueno

$q_R = \frac{2.5}{5}$

$q_B = \frac{2.5}{5}$

$L = \left(\frac{2.5}{5}\right)^2 \left(\frac{2.5}{5}\right)^3 = 0.031$

Cross-Entropy Loss Log-likelihood Perspective

Log-likelihood?

$$\prod_i q_i^{Np_i}$$

q_i = estimated probability of outcome i

p_i = empirical probability of outcome i in the training set

The log-likelihood, divided by N is:

$$\frac{1}{N}\log\prod_i q_i^{Np_i} = \sum_i p_i \log q_i =$$

PROPIEDADES DEL LOS LOGARITMOS

$\log_a(x \cdot y) = \log_a(x) + \log_a(y)$

$\log_a\left(\frac{x}{y}\right) = \log_a(x) - \log_a(y)$

$\log_a(x^n) = n \cdot \log_a(x)$

$\log_a\left(\frac{1}{x}\right) = -\log_a(x)$

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)}$$

Cross-Entropy Loss Log-likelihood Perspective

Log-likelihood?

$$\prod_i q_i^{Np_i}$$

q_i = estimated probability of outcome i

p_i = empirical probability of outcome i in the training set

Un momento 🤔

The log-likelihood, divided by N is:

$$\frac{1}{N}\log\prod_i q_i^{Np_i} = \sum_i p_i \log q_i =$$

PROPIEDADES DEL LOS LOGARITMOS

$$\log_a(x\cdot y)=\log_a(x)+\log_a(y)$$

$$\log_a\left(\frac{x}{y}\right)=\log_a(x)-\log_a(y)$$

$$\log_a(x^n)=n\cdot\log_a(x)$$

$$\log_a\left(\frac{1}{x}\right)=-\log_a(x)$$

$$\log_a(x)=\frac{\log_b(x)}{\log_b(a)}$$

Cross-Entropy Loss Log-likelihood Perspective

Log-likelihood?

$$\prod_i q_i^{Np_i}$$

q_i = estimated probability of outcome i

p_i = empirical probability of outcome i in the training set

The log-likelihood, divided by N is:

$$\frac{1}{N}\log\prod_i q_i^{Np_i} = \sum_i p_i \log q_i = -$$

Un momento 🤔

$$= -\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$ **CELoss**

PROPIEDADES DEL LOS LOGARITMOS

$\log_a(x \cdot y) = \log_a(x) + \log_a(y)$

$\log_a\left(\frac{x}{y}\right) = \log_a(x) - \log_a(y)$

$\log_a(x^n) = n \cdot \log_a(x)$

$\log_a\left(\frac{1}{x}\right) = -\log_a(x)$

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)}$$

Cross-Entropy Loss Log-likelihood Perspective

Log-likelihood?

$$\prod_i q_i^{Np_i}$$

q_i = estimated probability of outcome i
p_i = empirical probability of outcome i in the training set

The log-likelihood, divided by N is:

$$\frac{1}{N}\log\prod_i q_i^{Np_i} = \sum_i p_i \log q_i =$$

Un momento 🤔

$$= -\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$ **CELoss**

¿Qué nos falta?

PROPIEDADES DEL LOS LOGARITMOS

$\log_a(x \cdot y) = \log_a(x) + \log_a(y)$

$\log_a\left(\frac{x}{y}\right) = \log_a(x) - \log_a(y)$

$\log_a(x^n) = n \cdot \log_a(x)$

$\log_a\left(\frac{1}{x}\right) = -\log_a(x)$

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)}$$

Cross-Entropy Loss Log-likelihood Perspective

Log-likelihood?

$$\prod_i q_i^{Np_i}$$

q_i = estimated probability of outcome i
p_i = empirical probability of outcome i in the training set

The log-likelihood, divided by N is:

$$\frac{1}{N}\log\prod_i q_i^{Np_i} = \sum_i p_i \log q_i =$$

Un momento 🤔

$$= -\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$

¿Qué nos falta?⟶Negative!

PROPIEDADES DEL LOS LOGARITMOS

$\log_a(x \cdot y) = \log_a(x) + \log_a(y)$

$\log_a\left(\frac{x}{y}\right) = \log_a(x) - \log_a(y)$

$\log_a(x^n) = n \cdot \log_a(x)$

$\log_a\left(\frac{1}{x}\right) = -\log_a(x)$

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)}$$

Cross-Entropy Loss Log-likelihood Perspective

Log-likelihood?

$$\prod_i q_i^{Np_i}$$

q_i = estimated probability of outcome i

p_i = empirical probability of outcome i in the training set

The log-likelihood, divided by N is:

$$\frac{1}{N}\log\prod_i q_i^{Np_i} = \sum_i p_i \log q_i =$$

Un momento 🤔

$$= -\sum_{i=1}^{\#C} p_i \cdot ln(q_i)$$

¿Qué nos falta? ⟶ Negative!

PROPIEDADES DEL LOS LOGARITMOS

$\log_a(x \cdot y) = \log_a(x) + \log_a(y)$

$\log_a\left(\frac{x}{y}\right) = \log_a(x) - \log_a(y)$

$\log_a(x^n) = n \cdot \log_a(x)$

$\log_a\left(\frac{1}{x}\right) = -\log_a(x)$

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)}$$

NLLLoss

$$-\sum_i p_i \log q_i$$

Cross-Entropy Loss Log-likelihood Perspective

What is the difference?

En la práctica

CELoss NLLLoss

CrossEntropyLoss

The *target* that this criterion expects should contain either:

- Class indices in the range $[0, C)$ where C is the number of classes; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range). The unreduced (i.e. with `reduction` set to `'none'`) loss for this case can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

where x is the input, y is the target, w is the weight, C is the number of classes, and N spans the minibatch dimension as well as $d_1, ..., d_k$ for the *K*-dimensional case. If `reduction` is not `'none'` (default `'mean'`), then

NLLLoss

The *target* that this loss expects should be a class index in the range $[0, C-1]$ where *C = number of classes*; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range).

The unreduced (i.e. with `reduction` set to `'none'`) loss can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} x_{n,y_n}, \quad w_c = \text{weight}[c] \cdot 1\{c \neq \text{ignore\_index}\},$$

En la práctica

CELoss NLLLoss

CrossEntropyLoss

The *target* that this criterion expects should contain either:

- Class indices in the range $[0, C)$ where C is the number of classes; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range). The unreduced (i.e. with `reduction` set to `'none'`) loss for this case can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

where x is the input, y is the target, w is the weight, C is the number of classes, and N spans the minibatch dimension as well as $d_1, \dots, d_k$ for the *K*-dimensional case. If `reduction` is not `'none'` (default `'mean'`), then

L = Batch

y_n = target

w_{y_n} = Peso del target (Para datasets con desbalance de clases). Si está balanceado = 1.

x_n = input

NLLLoss

The *target* that this loss expects should be a class index in the range $[0, C-1]$ where *C = number of classes*; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range).

The unreduced (i.e. with `reduction` set to `'none'`) loss can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} x_{n,y_n}, \quad w_c = \text{weight}[c] \cdot 1\{c \neq \text{ignore\_index}\},$$

En la práctica

CELoss NLLLoss

CrossEntropyLoss

The *target* that this criterion expects should contain either:

- Class indices in the range $[0, C)$ where C is the number of classes; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range). The unreduced (i.e. with `reduction` set to `'none'`) loss for this case can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

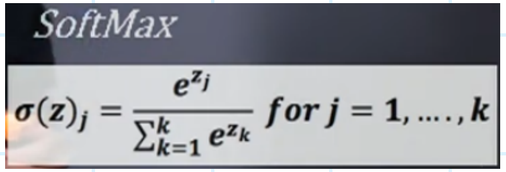
where x is the input, y is the target, w is the weight, C is the number of classes, and N spans the minibatch dimension as well as $d_1, \dots, d_k$ for the *K*-dimensional case. If `reduction` is not `'none'` (default `'mean'`), then

L = Batch

y_n = target

w_{y_n} = Peso del target (Para datasets con desbalance de clases). Si está balanceado = 1.

x_n = input



NLLLoss

The *target* that this loss expects should be a class index in the range $[0, C-1]$ where *C = number of classes*; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range).

The unreduced (i.e. with `reduction` set to `'none'`) loss can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} x_{n,y_n}, \quad w_c = \text{weight}[c] \cdot 1\{c \neq \text{ignore\_index}\},$$

En la práctica

CELoss NLLLoss

CrossEntropyLoss

The *target* that this criterion expects should contain either:

- Class indices in the range $[0, C)$ where C is the number of classes; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range). The unreduced (i.e. with `reduction` set to `'none'`) loss for this case can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

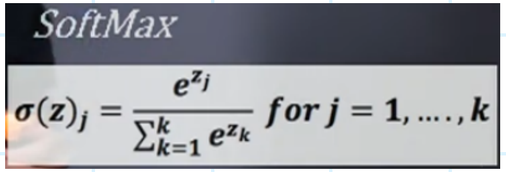
where x is the input, y is the target, w is the weight, C is the number of classes, and N spans the minibatch dimension as well as $d_1, \dots, d_k$ for the *K*-dimensional case. If `reduction` is not `'none'` (default `'mean'`), then

L = Batch

y_n = target

w_{y_n} = Peso del target (Para datasets con desbalance de clases). Si está balanceado = 1.

x_n = input



NLLLoss

The *target* that this loss expects should be a class index in the range $[0, C-1]$ where *C = number of classes*; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range).

The unreduced (i.e. with `reduction` set to `'none'`) loss can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} x_{n,y_n}, \quad w_c = \text{weight}[c] \cdot 1\{c \neq \text{ignore\_index}\},$$

Lo mismo, pero sin log-softmax

En la práctica

DEMO

CELoss NLLLoss

CrossEntropyLoss

The *target* that this criterion expects should contain either:

- Class indices in the range $[0, C)$ where C is the number of classes; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range). The unreduced (i.e. with `reduction` set to `'none'`) loss for this case can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

where x is the input, y is the target, w is the weight, C is the number of classes, and N spans the minibatch dimension as well as $d_1, \dots, d_k$ for the *K*-dimensional case. If `reduction` is not `'none'` (default `'mean'`), then

NLLLoss

The *target* that this loss expects should be a class index in the range $[0, C-1]$ where *C = number of classes*; if *ignore_index* is specified, this loss also accepts this class index (this index may not necessarily be in the class range).

The unreduced (i.e. with `reduction` set to `'none'`) loss can be described as:

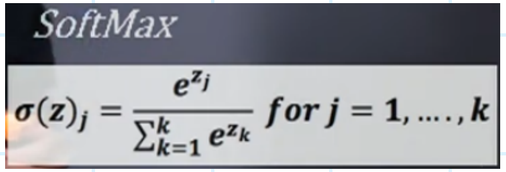
$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} x_{n,y_n}, \quad w_c = \text{weight}[c] \cdot 1\{c \neq \text{ignore\_index}\},$$

L = Batch

y_n = target

w_{y_n} = Peso del target (Para datasets con desbalance de clases). Si está balanceado = 1.

x_n = input



Lo mismo, pero sin log-softmax

Conclusiones

- **Si usas NLLLoss pon log-softmax en la última capa del modelo**
- **Mejor usa Cross Entropy Loss y te ahorras lo anterior**

Obtaining log-probabilities in a neural network is easily achieved by adding a *LogSoftmax* layer in the last layer of your network. You may use *CrossEntropyLoss* instead, if you prefer not to add an extra layer.

- **Si usas BCELoss pon sigmoid en la última capa del modelo**
- **Mejor usa BCEWithLogitsLoss que la aplica**

Conclusiones

- **Si usas NLLLoss pon log-softmax en la última capa del modelo**
- **Mejor usa Cross Entropy Loss y te ahorras lo anterior**

Obtaining log-probabilities in a neural network is easily achieved by adding a *LogSoftmax* layer in the last layer of your network. You may use *CrossEntropyLoss* instead, if you prefer not to add an extra layer.

- **Si usas BCELoss pon sigmoid en la última capa del modelo**
- **Mejor usa BCEWithLogitsLoss que la aplica**
- **No olvidar hacer print y guardar los valores de la loss para análisis**