

## Softmax regression (4 possible outputs)

$$\begin{aligned}
 \times z_1 &= \vec{w}_1 \cdot \vec{x} + b_1 & a_1 &= \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \\
 & & &= P(y = 1 | \vec{x}) \quad 0.30 \\
 \bigcirc z_2 &= \vec{w}_2 \cdot \vec{x} + b_2 & a_2 &= \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \\
 & & &= P(y = 2 | \vec{x}) \quad 0.20 \\
 \square z_3 &= \vec{w}_3 \cdot \vec{x} + b_3 & a_3 &= \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \\
 & & &= P(y = 3 | \vec{x}) \quad 0.15 \\
 \triangle z_4 &= \vec{w}_4 \cdot \vec{x} + b_4 & a_4 &= \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \\
 & & &= P(y = 4 | \vec{x}) \quad 0.35
 \end{aligned}$$

For a multiclass classification task that has 4 possible outputs, the sum of all the activations adds up to 1. For a multiclass classification task that has 3 possible outputs, the sum of all the activations should add up to ....

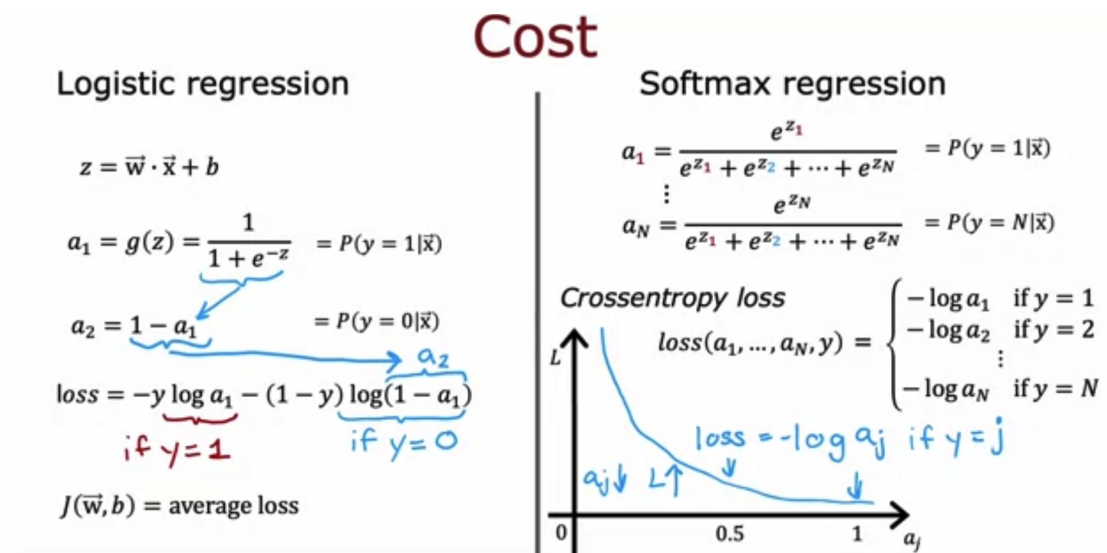
- ☐ Less than 1
- ☐ More than 1
- ☒ 1
- ☐ It will vary, depending on the input x.

✓ **Correcto**

Yes! The sum of all the softmax activations should add up to 1. One way to see this is that if  $e^{z_1} = 10$ ,  $e^{z_2} = 20$ ,  $e^{z_3} = 30$ , then the sum of  $a_1 + a_2 + a_3$  is equal to  $\frac{e^{z_1} + e^{z_2} + e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$  which is 1.

2.

1 / 1 punto



For multiclass classification, the cross entropy loss is used for training the model. If there are 4 possible classes for the output, and for a particular training example, the true class of the example is class 3 ( $y=3$ ), then what does the cross entropy loss simplify to? [Hint: This loss should get smaller when  $a_3$  gets larger.]

- ☐  $z_3/(z_1+z_2+z_3+z_4)$
- ☒  $-\log(a_3)$
- ☐  $z_3$
- ☐  $\frac{-\log(a_1) + -\log(a_2) + -\log(a_3) + -\log(a_4)}{4}$

✓ **Correcto**

Correct. When the true label is 3, then the cross entropy loss for that training example is just the negative of the log of the activation for the third neuron of the softmax. All other terms of the cross entropy loss equation ( $-\log(a_1)$ ,  $-\log(a_2)$ , and  $-\log(a_4)$ ) are ignored

3.

1 / 1 punto

## MNIST (more numerically accurate)

```
model    import tensorflow as tf
         from tensorflow.keras import Sequential
         from tensorflow.keras.layers import Dense
         model = Sequential([
             Dense(units=25, activation='relu')
             Dense(units=15, activation='relu')
             Dense(units=10, activation='linear') ])

loss     from tensorflow.keras.losses import
         SparseCategoricalCrossentropy

         model.compile(..., loss=SparseCategoricalCrossentropy(from_logits=True) )

fit      model.fit(X,Y,epochs=100)

predict  logits = model(X)
         f_x = tf.nn.softmax(logits)
```

For multiclass classification, the recommended way to implement softmax regression is to set `from_logits=True` in the loss function, and also to define the model's output layer with...

- ☒ a 'linear' activation
- ☐ a 'softmax' activation

✓ **Correcto**

Yes! Set the output as linear, because the loss function handles the calculation of the softmax with a more numerically stable method.