



Clasificación

- Regresión Logística
- Redes Neuronales



Bruno Bianchi

Regresión logística

Regresión logística: modelo discriminativo



Bayes

- Modelo generativo
- Calculamos $P(\text{clase}_i | \text{Datos})$ a partir de calcular el likelihood $\rightarrow P(\text{Datos} | \text{clase}_i)$
-

Regresión logística: modelo discriminativo

Bayes

- Modelo generativo
- Calculamos $P(\text{clase}_i | \text{Datos})$ a partir de calcular el likelihood $\rightarrow P(\text{Datos} | \text{clase}_i)$

Regresión Logística

- Se computa directamente $P(y|D)$, sin pasar por el Likelihood
- Puede usar features poco importantes para la generación



Regresión logística: componentes



Regresión logística: componentes



1. Representacion de features

$x^{(i)} = [x^{(i)}_1, x^{(i)}_2, \dots, x^{(i)}_n]$ -> asociado a una clase y

Regresión logística: componentes



1. Representacion de features

$x^{(i)} = [x^{(i)}_1, x^{(i)}_2, \dots, x^{(i)}_n]$ -> asociado a una clase y

2. Funcion de clasificacion, que compute \hat{y}

$f(x^{(i)}) = f([x^{(i)}_1, x^{(i)}_2, \dots, x^{(i)}_n]) = \hat{y}$

Regresión logística: componentes



1. Representacion de features

$x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]$ -> asociado a una clase y

2. Funcion de clasificacion, que compute \hat{y}

$f(x^{(i)}) = f([x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]) = \hat{y}$

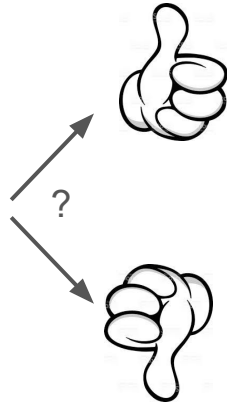
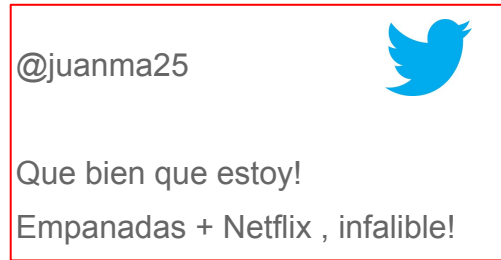
3. Función objetivo para el aprendizaje

Regresión logística: componentes



1. Representacion de features
 $x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]$ -> asociado a una clase y
2. Funcion de clasificacion, que compute \hat{y}
 $f(x^{(i)}) = f([x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]) = \hat{y}$
3. Función objetivo para el aprendizaje
4. Algoritmo para optimizar la función objetivo

1 - Representación de features



Feature	Valor
Cantidad de palabras positivas	1
Cantidad de palabras negativas	0
Cantidad de signos de exclamación	2
Cantidad de palabras	7
...	...

$$x^{(1)} = [1, 0, 2, 7, \dots, x_n^{(1)}] \rightarrow y = \text{positivo}$$

2 - Función de clasificación

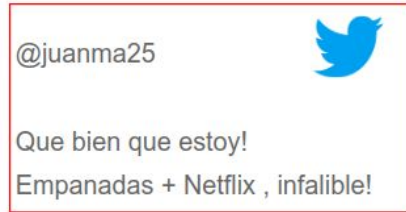


Funcion de clasificacion, que compute \hat{y}

$$f(x^{(i)}) = f([x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]) = \hat{y}$$

Dado un problema de clasificación binaria, necesitamos una función que tenga un resultado binarizado

2 - Función de clasificación

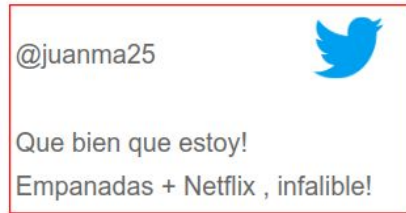


→ ¿Qué sentimiento expresa?

→ Positivo

→ Negativo

2 - Función de clasificación



→ ¿Qué sentimiento expresa?

→ Positivo

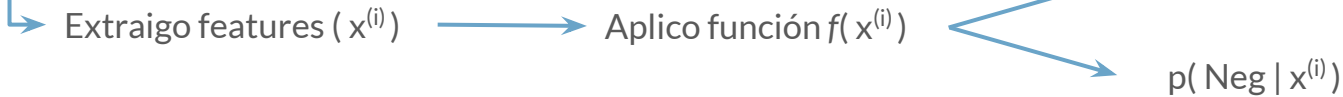
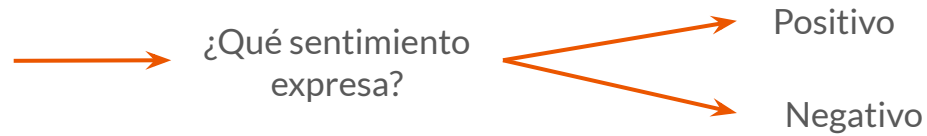
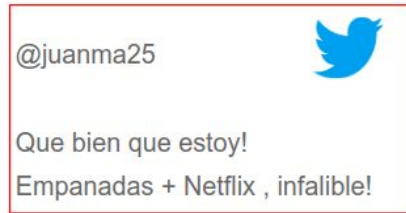
→ Negativo

→ Extraigo features ($x^{(i)}$) → Aplico función $f(x^{(i)})$

→ $p(\text{Pos} | x^{(i)})$

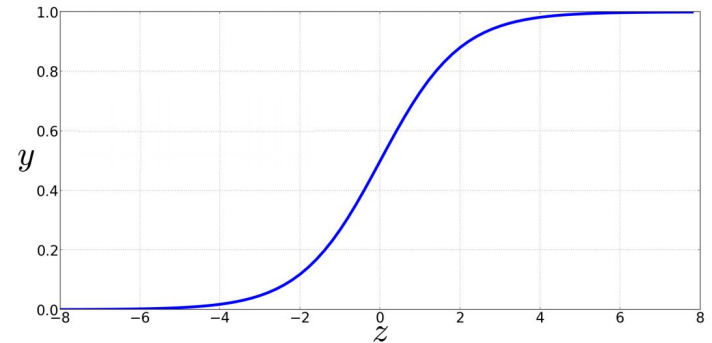
→ $p(\text{Neg} | x^{(i)})$

2 - Función de clasificación



$$x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]$$
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$
$$z = \left(\sum_{i=1}^n w_i x_i \right) + b$$

```
graph LR; A["x^(i) = [x_1^(i), x_2^(i), ..., x_n^(i)]"] --> B["sigma(z) = 1 / (1 + e^-z)"]; B --> C["z = (sum_{i=1}^n w_i x_i) + b"];
```



3 - Función objetivo (costo)



¿Cómo entrenamos el algoritmo ?

- Encontrar los pesos w_i que generen buenas predicciones
- Por cada instancia que predecimos, calculamos el costo (error) de esa predicción
- Modificamos los pesos w_i para minimizar el error

$L(\hat{y}, y)$ -> cuánta diferencia hay entre la predicción y el valor real

3 - Función objetivo (costo)



Cross - Entropy: entropía cruzada

$$L_{CE}(\hat{y}, y) = - [y \log (\hat{y}) + (1-y) \log(1-\hat{y})]$$

3 - Función objetivo (costo)



Cross - Entropy: entropía cruzada

$$L_{CE}(\hat{y}, y) = - [y \log (\hat{y}) + (1-y) \log(1-\hat{y})]$$

Si $y = 0$

$$L_{CE}(\hat{y}, y) = - \log(1 - \hat{y})$$

Si $y = 1$

$$L_{CE}(\hat{y}, y) = - \log (\hat{y})$$

4 - Algoritmo de optimización



¿ Cómo minimizamos la pérdida (costo) ?

$$L_{CE}(\hat{y}, y) = - [y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$$

$$L_{CE}(w, b) = -[y \log \sigma(w \cdot x + b) + (1-y) \log(1 - \sigma(w \cdot x + b))]$$

4 - Algoritmo de optimización

¿ Cómo minimizamos la pérdida (costo) ?

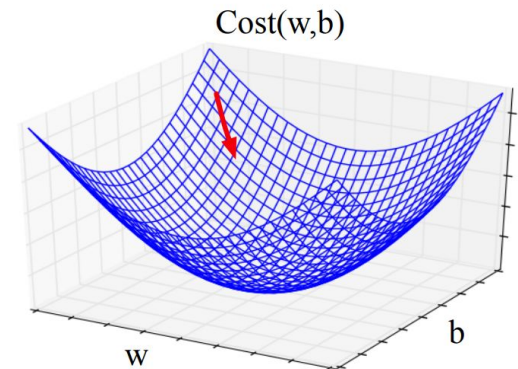
$$L_{CE}(\hat{y}, y) = - [y \log (\hat{y}) + (1-y) \log(1- \hat{y})]$$

$$L_{CE}(w, b) = -[y \log \sigma(w \cdot x + b) + (1-y) \log(1- \sigma(w \cdot x + b))]$$

Tenemos que modificar los pesos y bias para minimizar la pérdida

Derivamos la función de costo y buscamos la dirección en la cual baja la pérdida

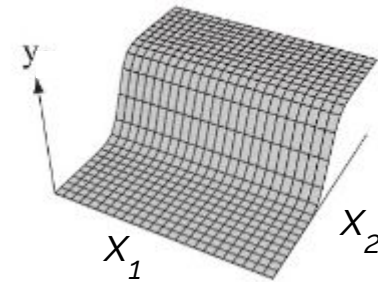
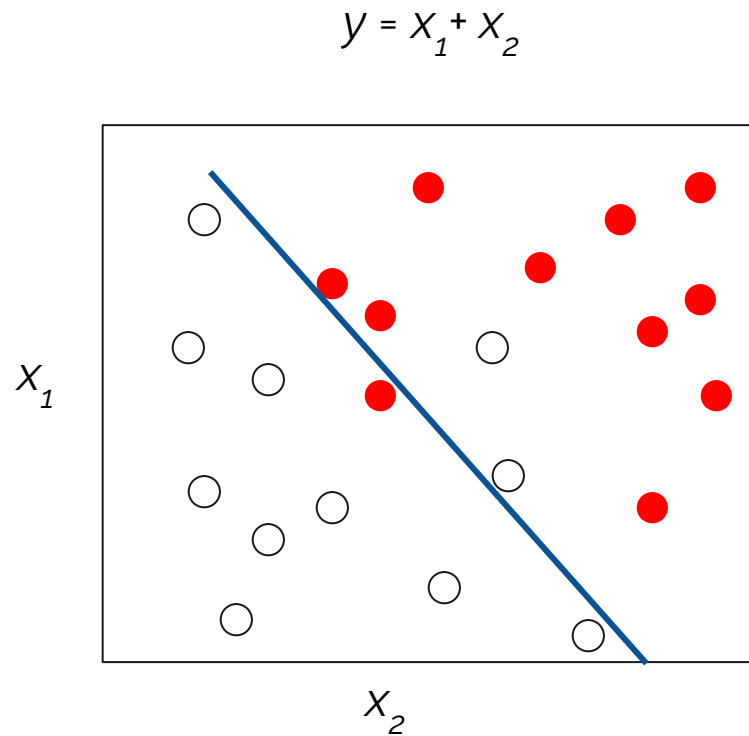
Cross-entropy con la función sigmoidea es derivable y convexa



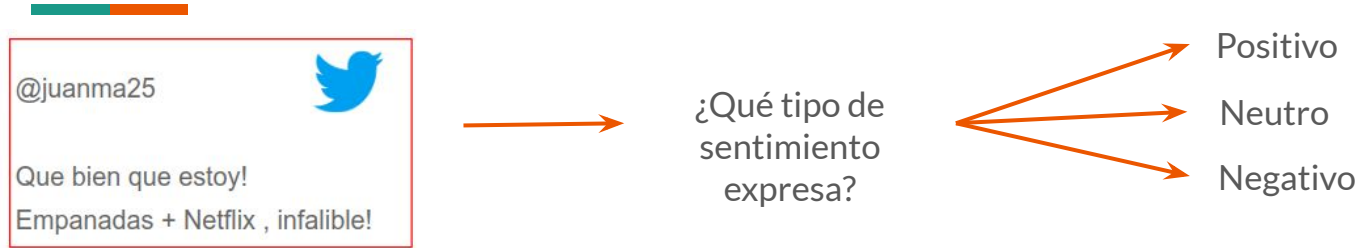
Regresión logística, discriminación lineal

$$y = X_1 + X_2$$

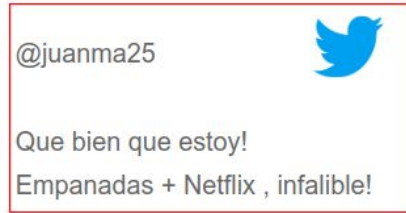
Regresión logística, discriminación lineal



Clasificación multiclase



Clasificación multiclase



¿Qué tipo de sentimiento expresa?



Positivo
Neutro
Negativo



Extraigo features ($x^{(i)}$)
(por cada clase)

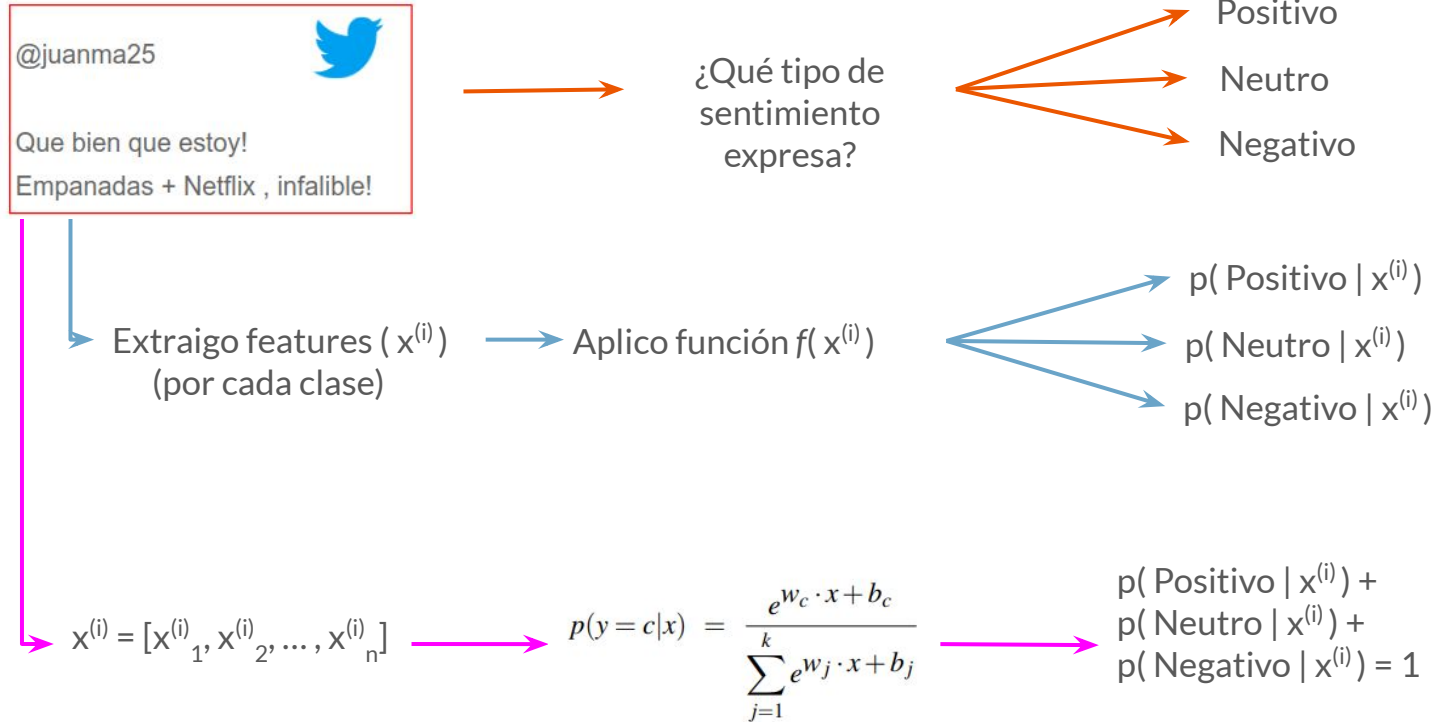


Aplico función $f(x^{(i)})$



$p(\text{Positivo} | x^{(i)})$
 $p(\text{Neutro} | x^{(i)})$
 $p(\text{Negativo} | x^{(i)})$

Clasificación multiclase



Regularización

Regularización



Si los pesos de los features se vuelven muy grandes, probablemente estemos overfitteando

Regularización: penalizar los pesos altos a la hora de calcular el costo

$$L_{CE}(w, b) = -[y \log \sigma(w \cdot x + b) + (1-y) \log(1 - \sigma(w \cdot x + b))] + \lambda R_{(w,b)}$$

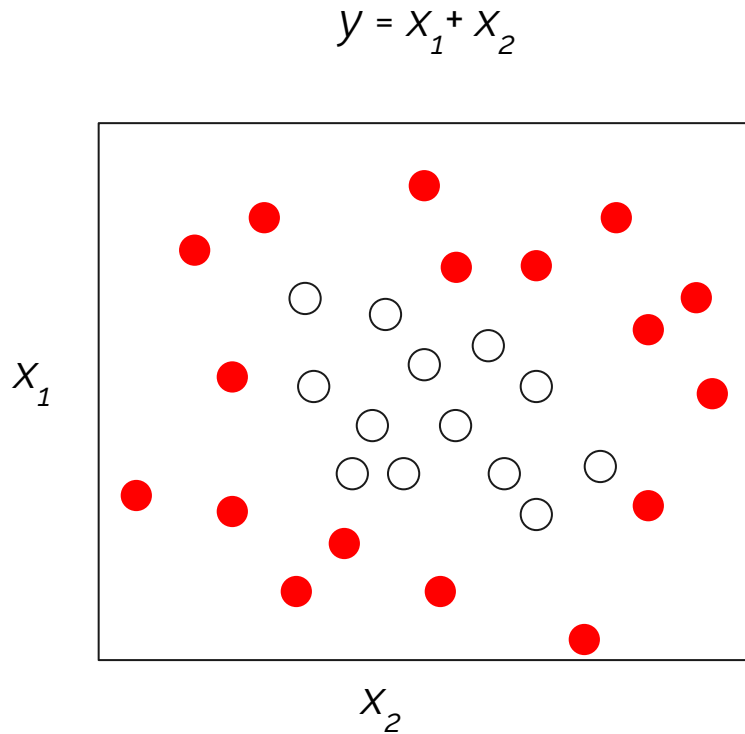
$$R_{(w,b)} = \sum |w_i|^N + |b|^N$$

$N = 1 \rightarrow$ Lasso

$N = 2 \rightarrow$ Ridge

Redes neuronales

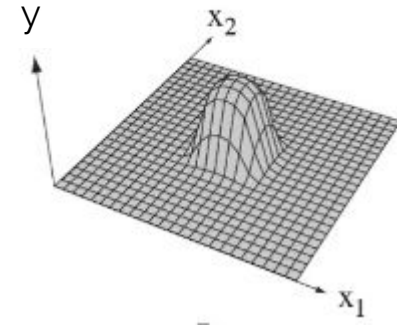
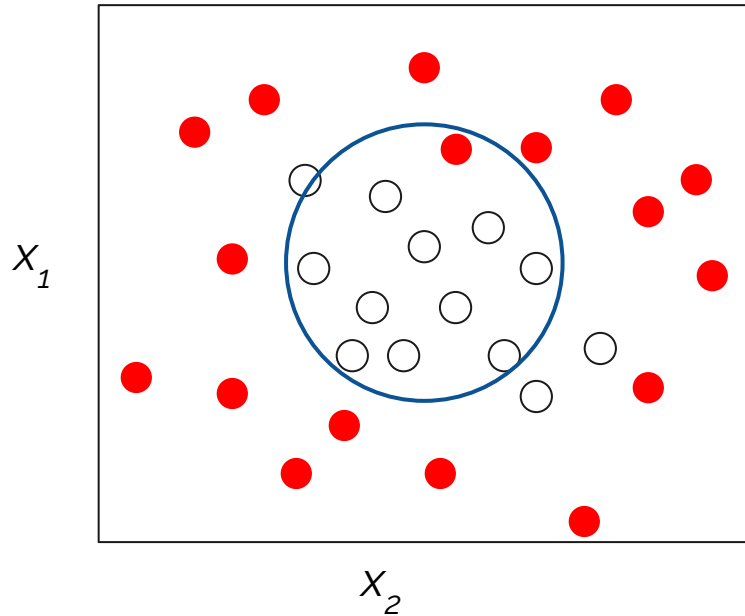
¿Y si tenemos un problema más complejo?



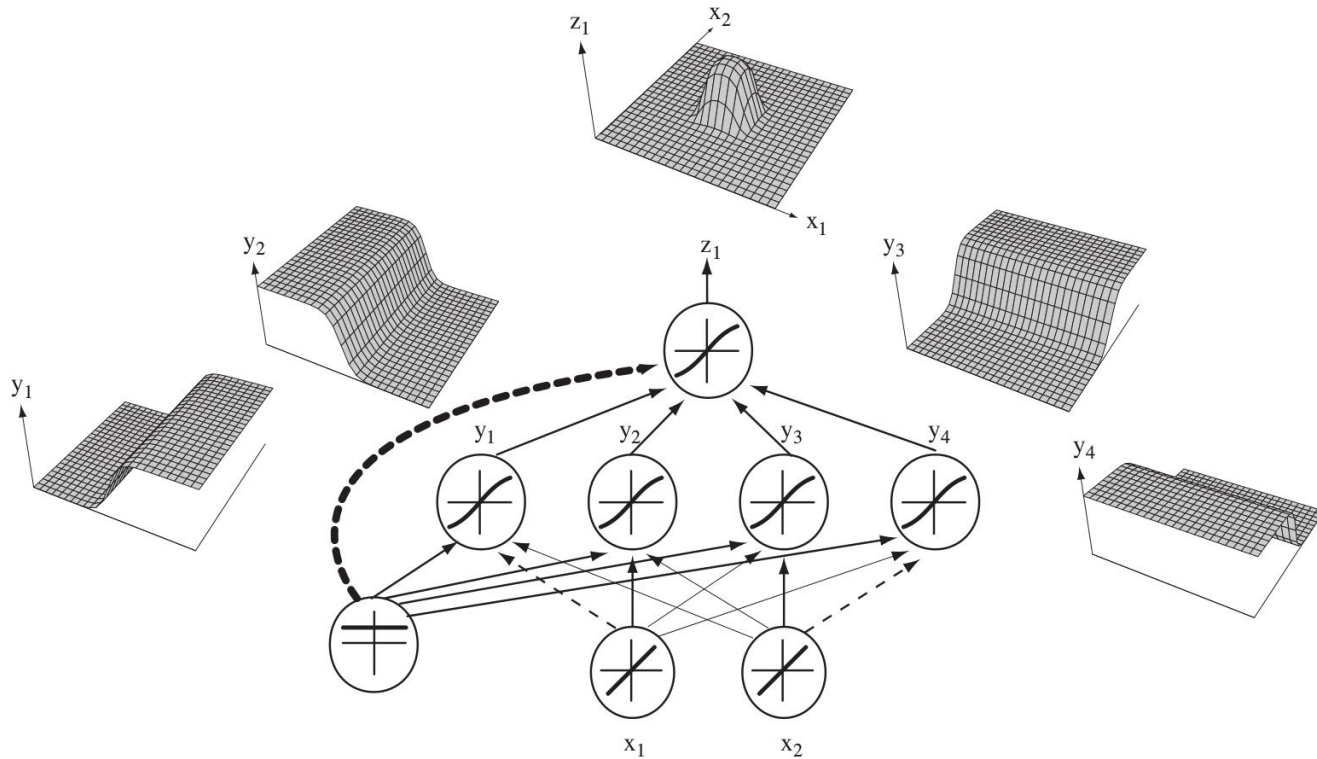
¿Y si tenemos un problema más complejo?



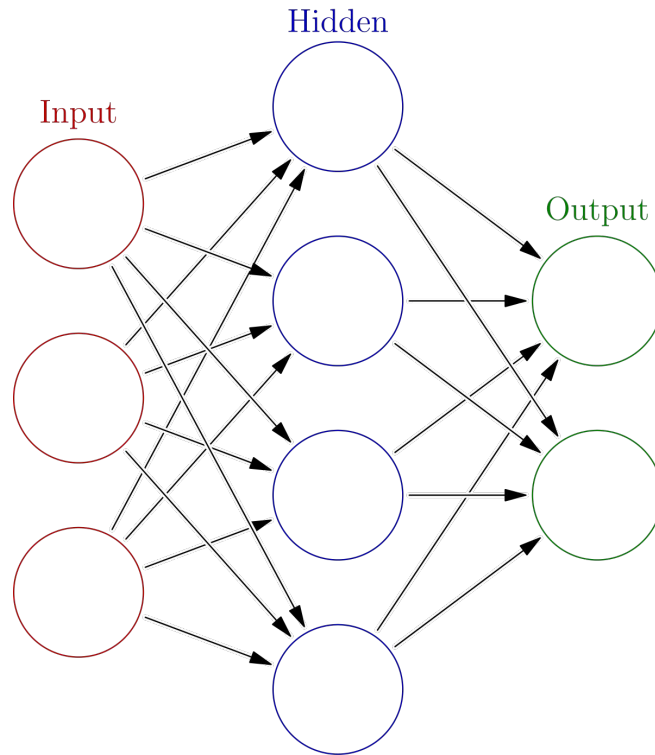
$$y = x_1 + x_2$$



¿Y si tenemos un problema más complejo?

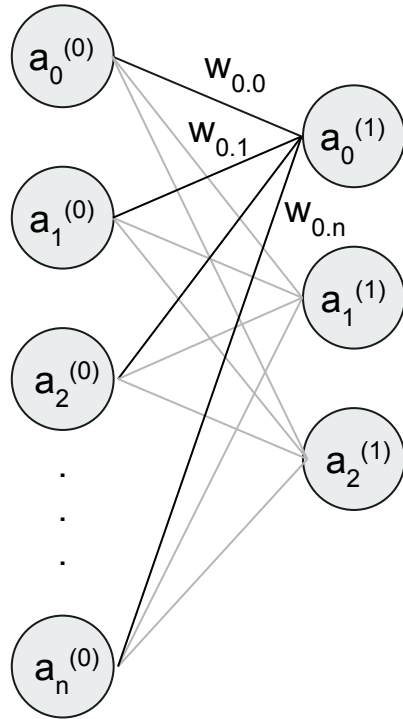


Estructura básica



Activación sigmoidea

Input layer Hidden 1

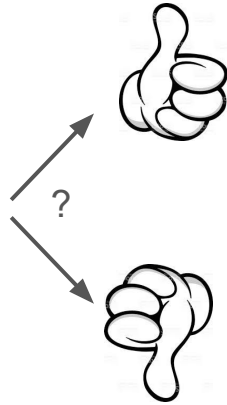
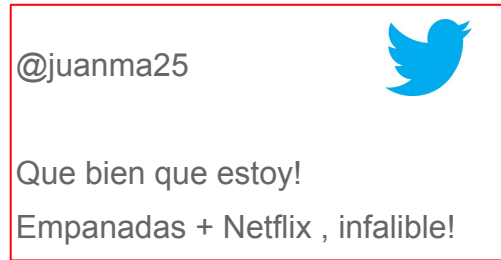


$$a_0^{(1)} = \sigma (w_{0.0} a_0^{(0)} + w_{0.1} a_1^{(0)} + \dots + w_{0.n} a_n^{(0)} + b_0)$$

$$a_1^{(1)} = \sigma (w_{1.0} a_0^{(0)} + w_{1.1} a_1^{(0)} + \dots + w_{1.n} a_n^{(0)} + b_0)$$

$$a_2^{(1)} = \sigma (w_{2.0} a_0^{(0)} + w_{2.1} a_1^{(0)} + \dots + w_{2.n} a_n^{(0)} + b_0)$$

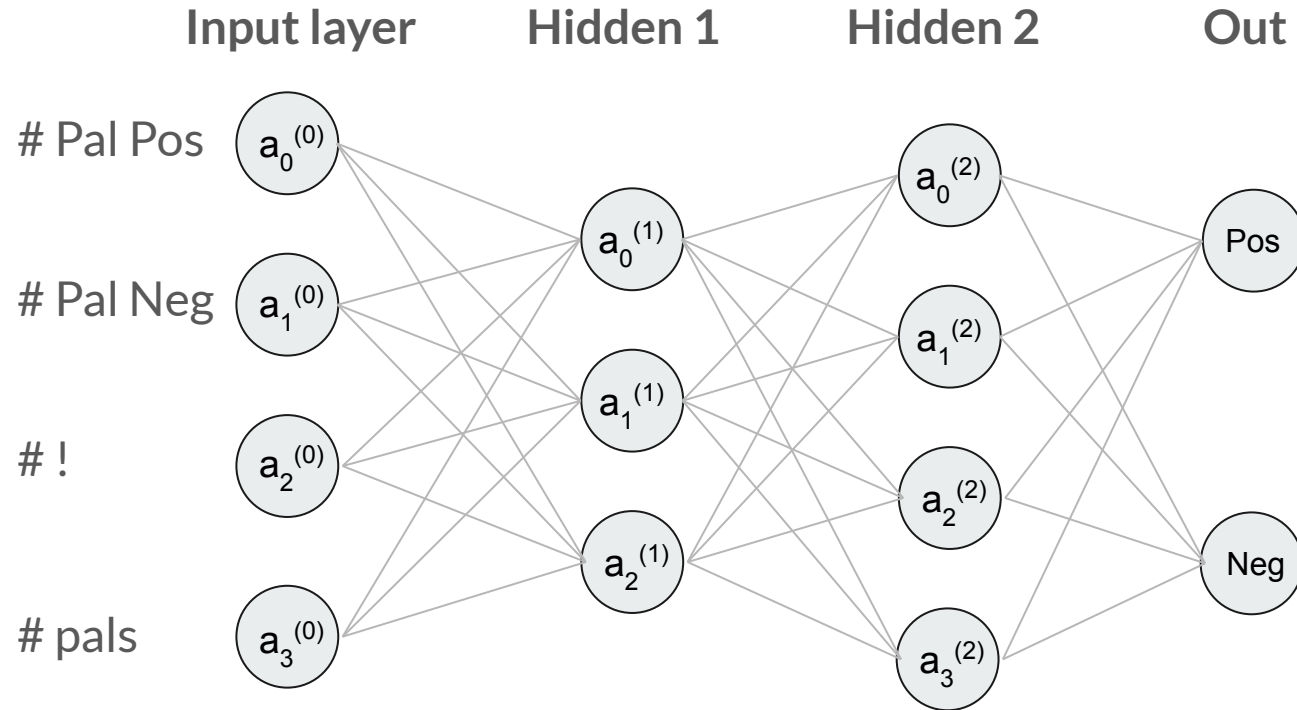
Redes Neuronales



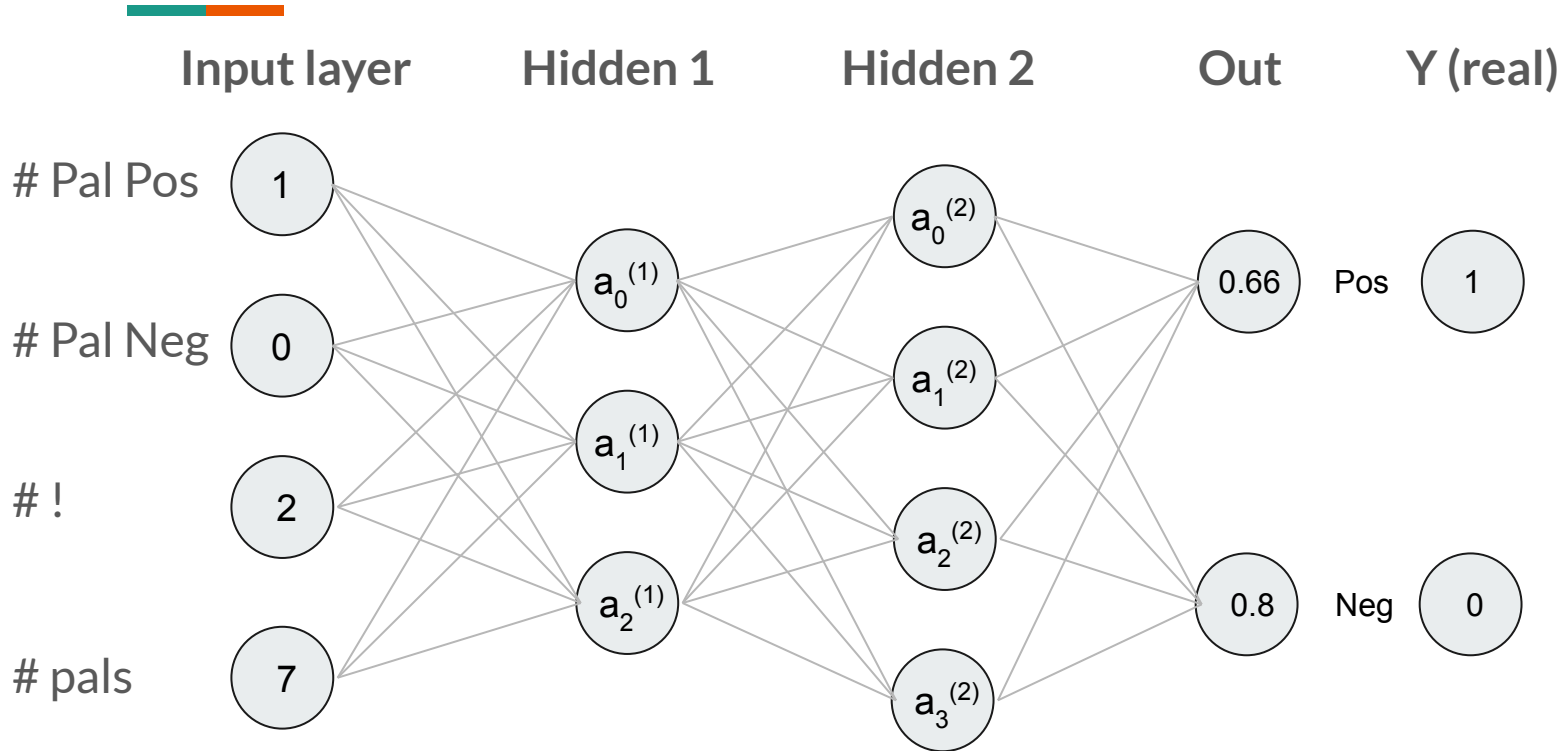
Feature	Valor
Cantidad de palabras positivas	1
Cantidad de palabras negativas	0
Cantidad de signos de exclamación	2
Cantidad de palabras	7
...	...

$$x^{(1)} = [1, 0, 2, 7, \dots, x_n^{(1)}] \rightarrow y = \text{positivo}$$

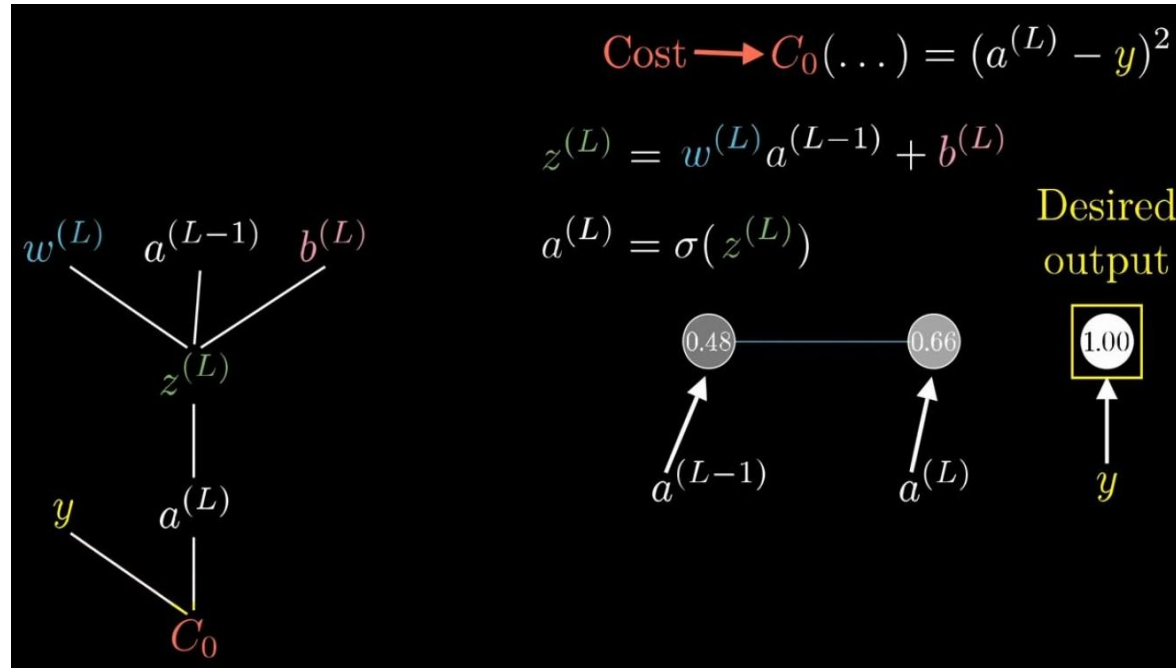
Entrenamiento (backpropagation)



Entrenamiento (backpropagation)



Entrenamiento (backpropagation)



$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = a^{(L-1)} \sigma'(z^{(L)}) 2(a^{(L)} - y)$$

Entrenamiento (backpropagation)



Finalmente, para actualizar los valores de w hago:

$$w_i^+ = w_i - \eta * \frac{\partial C}{\partial w_i}$$

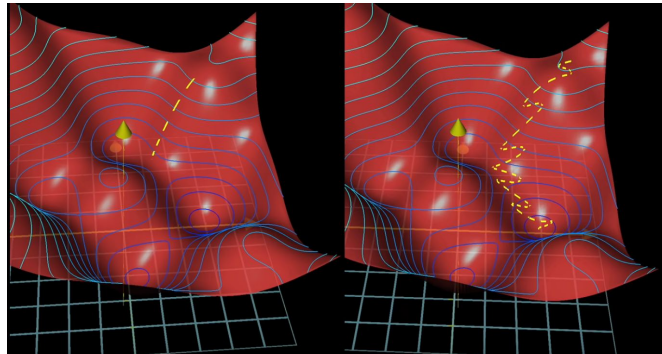
donde η es el “learning rate”

Stochastic Gradient Descent

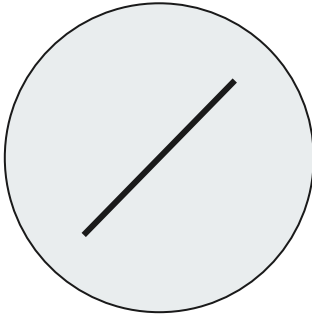
$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = a^{(L-1)} \sigma'(z^{(L)}) 2(a^{(L)} - y)$$

Hacer esta cuenta para cada instancia es muy costoso

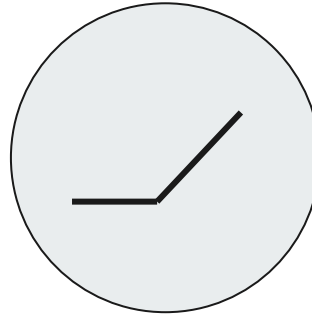
- Se entrenan mini-batches
- Se calcula el costo medio
- Se actualiza sobre con gradiente del mini-batch



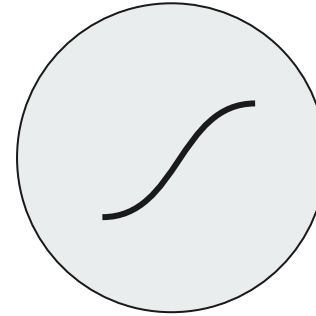
Tipos de neuronas



Lineal
 $y = w a + b$



ReLu
 $y = \begin{cases} 0 & \text{si } a < 0 \\ w a + b & \text{si } a \geq 0 \end{cases}$



Tanh
 $y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

Pre-procesamiento

Pre-procesamiento: Term-frequency matrix:



- “Más queso siempre es mejor, siempre”
- “Cerró la casa del queso y sus empleados tomaron el restaurant. Los empleados no fueron indemnizados”

	más	queso	siempre	mejor	cerró	casa	empleados	tomaron	restaurant	fueron	indemnizados
d1	1	1	2	1	0	0	0	0	0	0	0
d2	0	1	0	0	1	1	2	1	1	1	1

Notar que **queso** tiene igual peso en ambos documentos

Normalizacion



	más	queso	siempre	mejor	cerró	casa	empleados	tomaron	restaurant	fueron	indemnizados
d1	1	1	2	1	0	0	0	0	0	0	0
d2	0	1	0	0	1	1	2	1	1	1	1

$$d_1 = (1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0)$$

$$d_2 = (0, 1, 0, 0, 1, 1, 2, 1, 1, 1, 1)$$

Normalizacion



	más	queso	siempre	mejor	cerró	casa	empleados	tomaron	restaurant	fueron	indemnizados
d1	1	1	2	1	0	0	0	0	0	0	0
d2	0	1	0	0	1	1	2	1	1	1	1

$$d_1 = (1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0)$$

$$d_2 = (0, 1, 0, 0, 1, 1, 2, 1, 1, 1, 1)$$

Normalización Ln $\longrightarrow \hat{d} = \frac{d}{\|d\|_n} = \frac{d}{\sqrt[n]{\sum_i d_i^n}}$ Norma L2 (euclidea)

$$d_1 = \frac{(1,1,2,1,0,0,0,0,0,0,0)}{\sqrt{1+1+4+1}} = (0.38, 0.38, 0.76, 0.38, 0, 0, 0, 0, 0, 0, 0)$$

$$d_2 = \frac{(0,1,0,0,1,1,2,1,1,1,1)}{\sqrt{1+1+1+4+1+1+1+1+1}} = (0, 0.3, 0, 0, 0.3, 0.3, 0.6, 0.3, 0.3, 0.3, 0.3)$$

Term Frequency weight

- 
- “Más queso siempre es mejor, siempre”

Estaría bueno resaltar las palabras más significativas

¿Cómo cuantifico cuán significativa es una palabra?

Term Frequency weight




$$\text{tf-idf}(t, d) = \text{tf}(t, d) \cdot \text{idf}(t)$$

$$\text{idf}(t) = \log \frac{|D|}{|\{d:t \in d\}|}$$

Número de documentos en el set de entrenamiento

Número de documentos en los que aparece el término t

Term Frequency weight


$$\text{tf-idf}(t, d) = \text{tf}(t, d) \cdot \text{idf}(t)$$

$$\text{idf}(t) = \log \frac{|D|}{|\{d:t \in d\}|} + 1$$

Número de documentos en el set de entrenamiento

Evita $\text{idf}=0$ para los términos que están en todos los documentos

Número de documentos en los que aparece el término t

Term Frequency weight



Número de documentos en el set de entrenamiento

$$\text{idf}(t) = \log \frac{|D|}{|\{d:t \in d\}|} + 1$$

Número de documentos en los que aparece el término t

Con add-1 smoothing

$$\text{idf}(t) = \log \frac{1+|D|}{1+|\{d:t \in d\}|} + 1$$

Equivalente a agregar un documento que contenga todas las palabras. Esto evita el “cero” en el denominador

Term Frequency weight

- “Más queso siempre es mejor, siempre”

$$d_{test} = (1, 1, 2, 1, 0, 0, 0, 0, 0) \leftarrow \text{tf}$$

Term	tf	idf	tf-idf
más	1	1.1	1.1
queso	1	3	3
siempre	2	1.1	2.2
mejor	1	1.4	1.4
pomelo	0	4.1	0

Term Frequency weight

➤ “Más queso siempre es mejor, siempre”

Term	tf	idf	tf-idf
más	1	1.1	1.1
queso	1	3	3
siempre	2	1.1	2.2
mejor	1	1.4	1.4
pomelo	0	4.1	0

$$d_{test} = (1, 1, 2, 1, 0, 0, 0, 0, 0) \leftarrow \text{tf}$$

$$d_{test} = (1.1, 3, 2.2, 1.4, 0, 0, 0, 0, 0) \leftarrow \text{tf-idf}$$

$$d_{test} = (0.27, 0.73, 0.53, 0.34, 0, 0, 0, 0, 0) \leftarrow \text{tf-idf} + \text{norm}$$