



Aprendizaje Automático Profundo (Deep Learning)

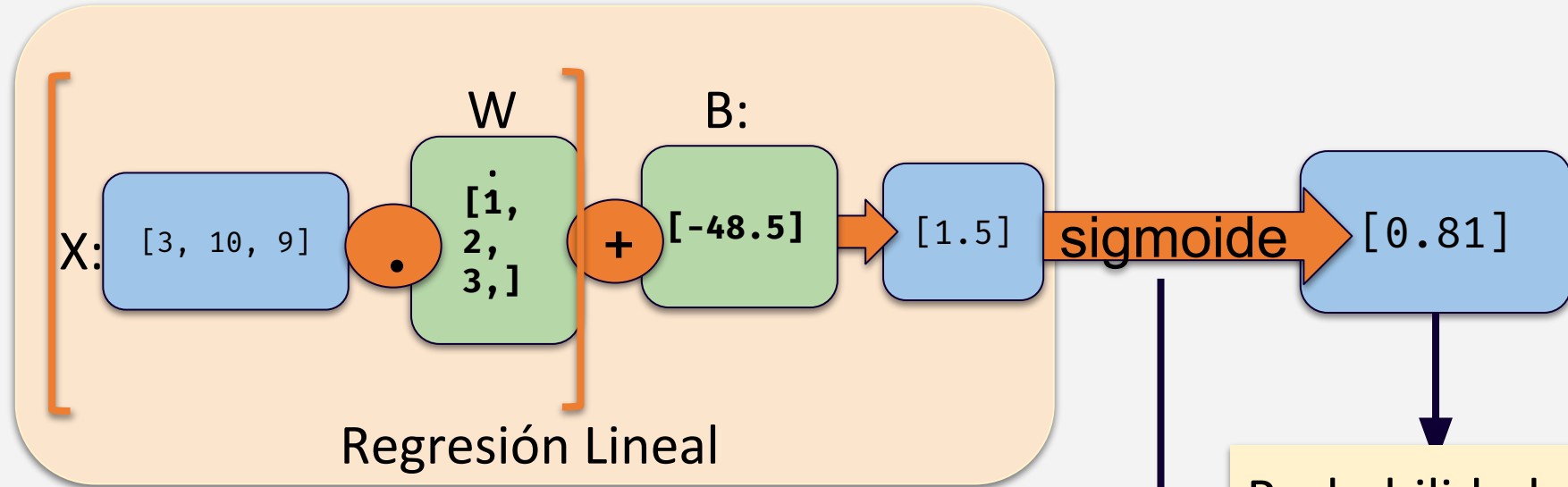
Dr. Facundo Quiroga - Dr. Franco Ronchetti



Regresión Logística con Múltiples Variables de Salida

Regresión Logística (dos clases)

Estudio	Edad	Promedio	Aprobó (y)
2	24	4	0
5	22	3	1
7	25	4	0
9	20	7	1
10	19	4	0
11	20	3	0
13,4	21	5	1
14	20	3	0



¿Si hay N clases?

$$s(x) = 1 / (1 + \exp(-x))$$

```
def sigmoide(x):  
    d = 1 + np.exp(-x)  
    return 1/d
```

Probabilidad de aprobar.
Si > 0.5 ,
clasificar como
"aprobado"

Regresión Logística Múltiple

Múltiples opciones
=> predicciones independientes

Variables (x)			Carrera (y)		
Estudio	Edad	Promedio	P1	P2	P3
2	24	4	0	1	1
5	22	3	1	1	1
7	25	4	0	1	0
9	20	7	1	0	1
10	19	4	0	1	0
11	20	3	1	1	1
13,4	21	5	1	0	1
14	20	3	0	1	0

Clases mutuamente exclusivas
=> Problema de **clasificación**

Variables (x)			Carrera (y)		
Estudio	Edad	Promedio	LS	LI	IC
2	24	4	1	0	0
5	22	3	0	1	0
7	25	4	0	1	0
9	20	7	0	0	1
10	19	4	0	0	0
11	20	3	0	0	1
13,4	21	5	0	1	0
14	20	3	1	0	0

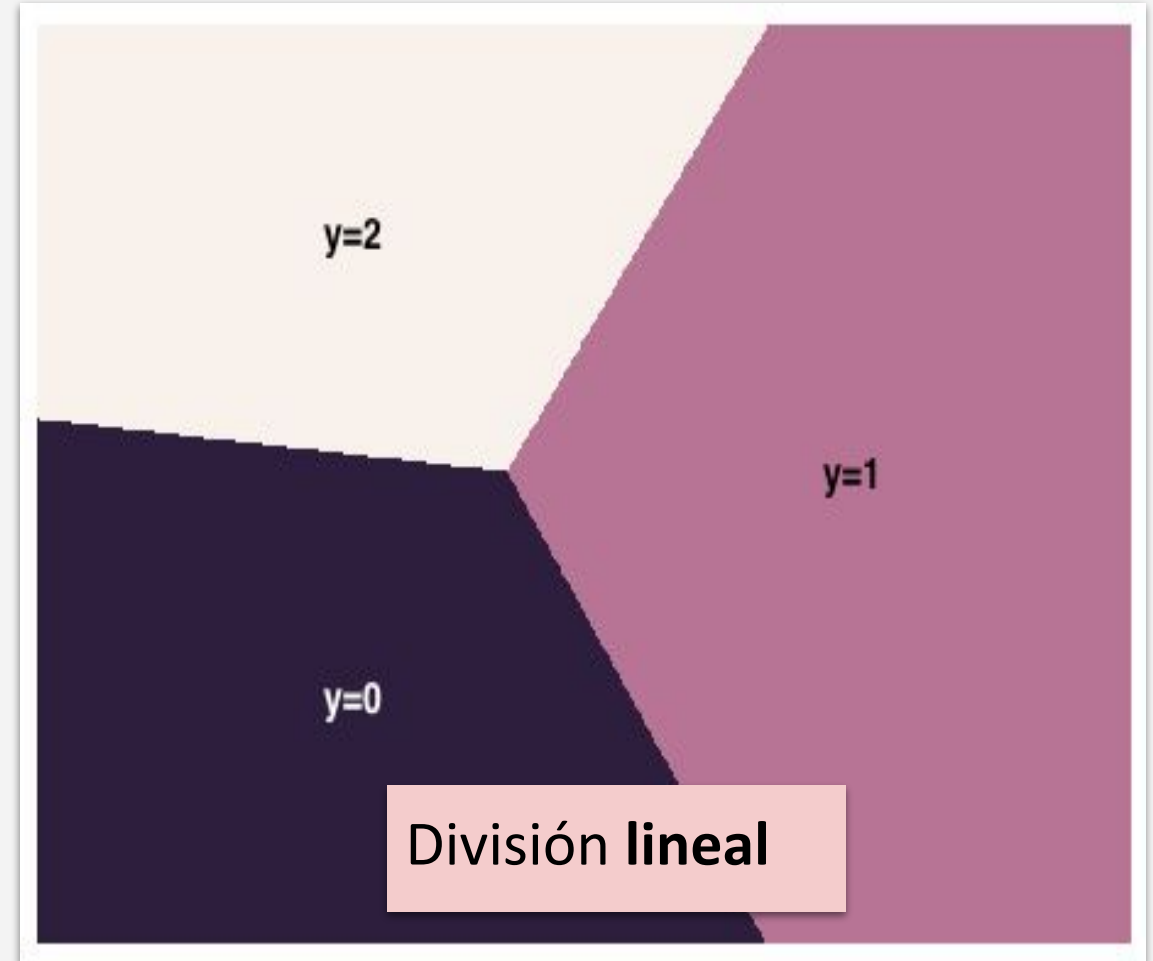
En esta materia:
**problemas de
clasificación**

Regresión Logística Múltiple - Intuición

Un hiperplano por cada clase



Dividen el espacio en regiones por clase



Regresión Logística Múltiple - Codificación

Codificación “One Hot”. Cada clase con su columna. Tamaño de y : $N \times C$

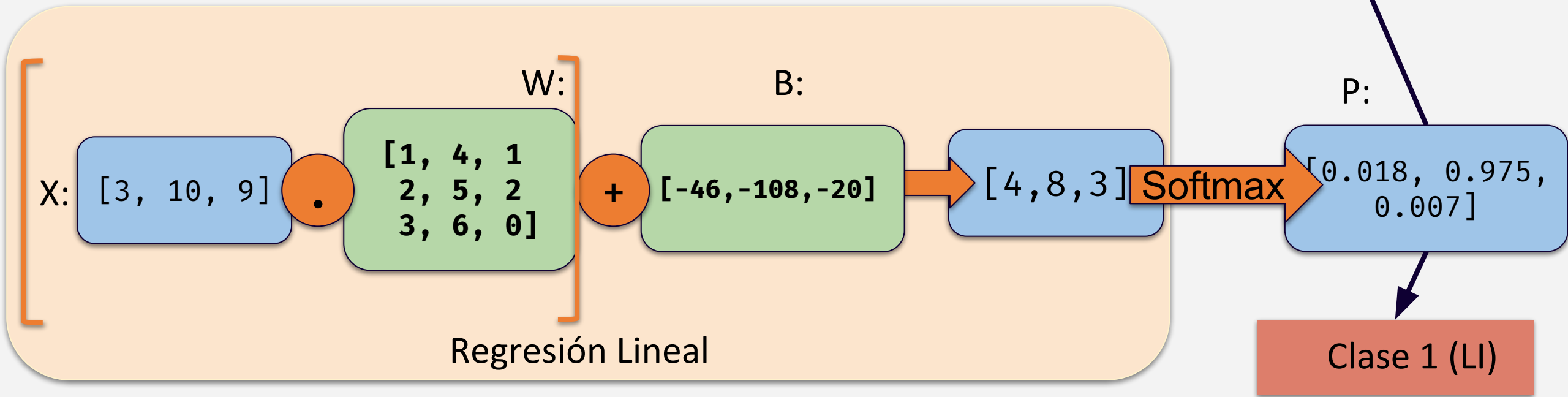
Variables (x)			Carrera (y)		
Estudio	Edad	Promedio	LS	LI	IC
2	24	4	1	0	0
5	22	3	0	1	0
7	25	4	0	1	0
9	20	7	0	0	1
10	19	4	1	0	0
11	20	3	0	0	1
13,4	21	5	0	1	0
14	20	3	1	0	0

Codificación con etiquetas. Cada clase con su ID. Tamaño de y : $N \times 1$

Variables (x)			Carrera (y)
Estudio	Edad	Promedio	LS = 0, LI = 1, IC = 2
2	24	4	0
5	22	3	1
7	25	4	1
9	20	7	2
10	19	4	0
11	20	3	2
13,4	21	5	1
14	20	3	0

Softmax

- Similar a **Regresión Lineal** pero para predecir probabilidades
- Función **softmax** aplicada a la salida
 - Genera **distribución de probabilidad $P(y)$**
- **Clasificación de N clases**



Softmax

- Función **softmax**
 - Convierte salida de Regresión Lineal en distribución de probabilidades de c/ clase

¿Por qué e^x ?

- Resultado > 0
- Exponencial, acentúa los valores altos
- Podría ser con otra base

Y:

[4, 8, 3]

Softmax

1) Cálculo del vector E

$$E = [e^4, e^8, e^3] = [54.59, 2980.95, 20.08]$$

2) Cálculo del valor N

$$N = e^4 + e^8 + e^3 = 54.59 + 2980.95 + 20.08 = 3055.63$$

3) Cálculo del vector P de probabilidades

$$\begin{aligned} P &= E / N = [e^4/N, e^8/N, e^3/N] \\ &= [54.59/3055.63, 2980.95/3055.63, 20.08/3055.63] \end{aligned}$$

P:

[0.018, 0.975,
0.007]

Softmax

Y:

[4, 8, 3]

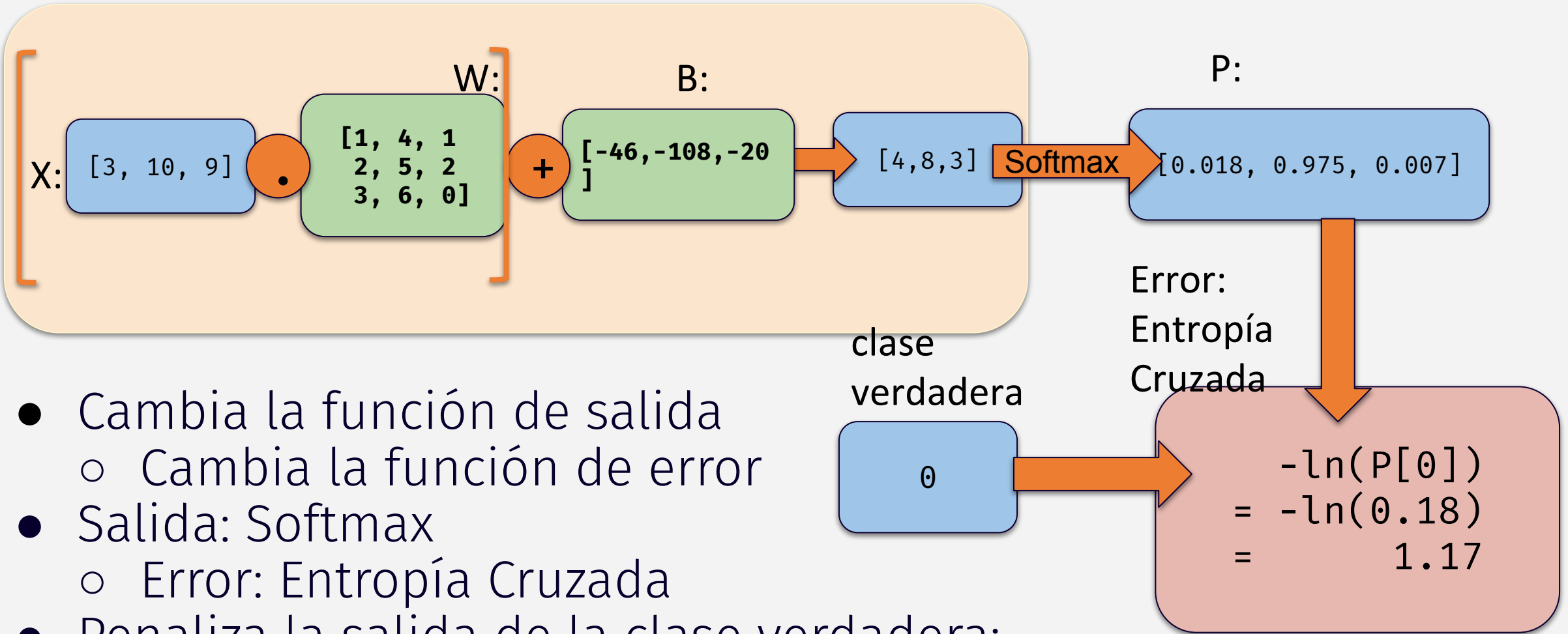
```
def softmax(Y):  
    n=len(Y)  
    E=np.zeros_like(Y)  
    for i in range(n):  
        E[i] = np.exp(Y[i])  
    N=0  
    for i in range(n):  
        N=N+E[i]  
    P=np.zeros_like(Y)  
    for i in range(n):  
        P[i]=E[i]/N  
    return P
```

P:

[0.018, 0.975,
0.007]

Versión vectorial:
def softmax(Y):
 E=np.exp(Y)
 N=np.sum(E)
 return E/N

Función de error: Entropía Cruzada



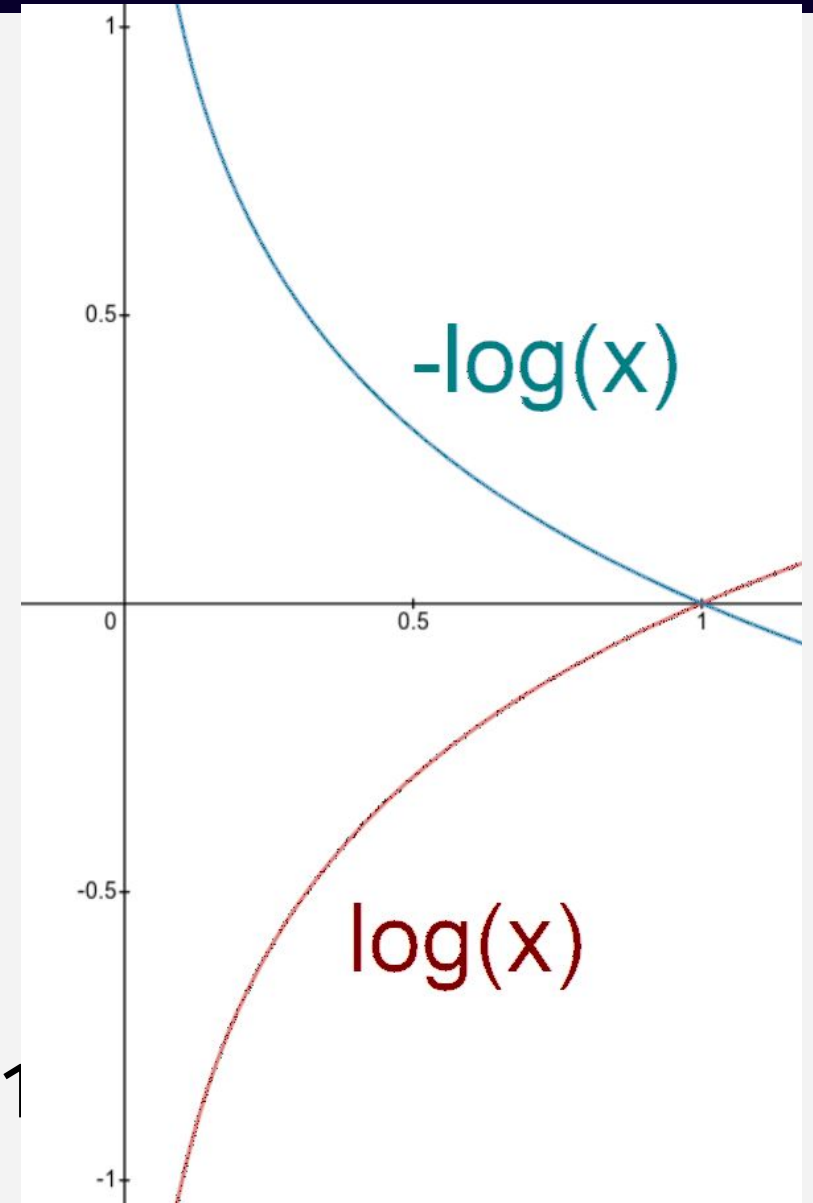
- Cambia la función de salida
 - Cambia la función de error
- Salida: Softmax
 - Error: Entropía Cruzada
- Penaliza la salida de la clase verdadera:
 - $-\ln(P[\text{clase_verdadera}])$

Repaso Logaritmo

- Funciones $\ln(x)$ y $-\ln(x)$
 - Cuando $0 \leq x \leq 1$
- Si x tiende a 1
 - $\ln(x)$ tiende a 0
 - $-\ln(x)$ tiende a 0
- Si x tiende a 0
 - $\ln(x)$ tiende a $-\infty$
 - $-\ln(x)$ tiende a $+\infty$

Entonces

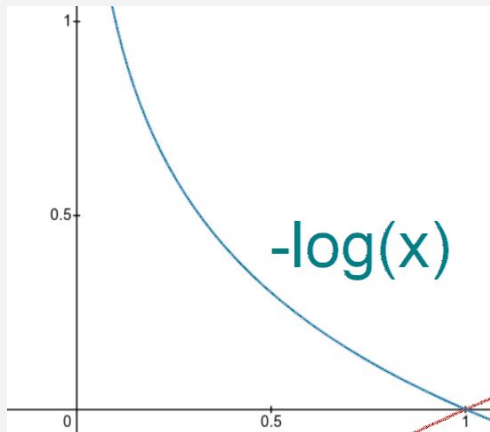
- $-\ln(P[\text{clase_verdadera}])$
 - Penaliza si $P[\text{clase_verdadera}] = 0$
 - NO penaliza si $P[\text{clase_verdadera}] = 1$



Función de error - Entropía Cruzada

Función de error = Entropía Cruzada = $-\ln(P[\text{clase_verdadera}])$

Variables			Carrera	xW+b			P(Carrera)			E	
Estudio	Edad	Promedio	LS = 0, LI = 1, IC = 2	LS	LI	IC	LS	LI	IC		
2	24	4	0	46	50	50.2	0.01	0.45	0.55	$-\log(0.01)$	4.61
14	20	3	0	69	66	66.1	0.91	0.05	0.05	$-\log(0.91)$	0.09



- $-\ln(P[\text{clase_verdadera}])$
 - Penaliza mucho si $P[\text{clase_verdadera}] \sim 0$
 - Penaliza poco si $P[\text{clase_verdadera}] \sim 1$

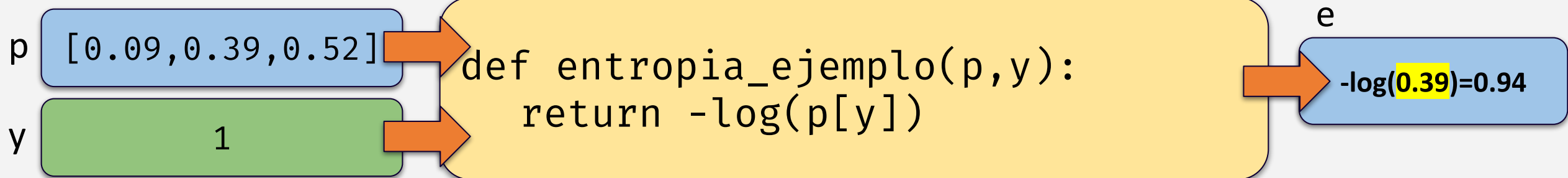
Función de error - Entropía Cruzada

Función de error = Entropía Cruzada Promedio = E.mean()

Variables			Carrera	xW+b			P(Carrera)			E	
Estudio	Edad	Promedio	LS = 0, LI = 1, IC = 2	LS	LI	IC	LS	LI	IC		
2	24	4	0	46	50	50.2	0.01	0.45	0.55	-log(0.01)	4.61
5	22	3	1	49.5	51	51.3	0.09	0.39	0.52	-log(0.39)	0.94
7	25	4	1	60	61.5	61.8	0.09	0.39	0.52	-log(0.39)	1.20
9	20	7	2	60.5	64	62.9	0.02	0.73	0.24	-log(0.24)	1.43
10	19	4	0	58.5	58.5	58.2	0.36	0.36	0.27	-log(0.36)	1.35
11	20	3	2	61.5	60	60.1	0.68	0.15	0.17	-log(0.17)	0.39
13,4	21	5	1	71	70.3	69.9	0.55	0.27	0.18	-log(0.55)	1.31
14	20	3	0	69	66	66.1	0.91	0.05	0.05	-log(0.91)	0.09

Error Promedio = 1.41

Función de error - Entropía Cruzada



Y	P		
LS = 0, LI = 1, IC = 2	LS	LI	IC
0	0.01	0.45	0.55
1	0.09	0.39	0.52
1	0.09	0.39	0.52
2	0.02	0.73	0.24
0	0.36	0.36	0.27
2	0.68	0.15	0.17
1	0.55	0.27	0.18
0	0.91	0.05	0.05

```
def entropia_promedio(P, Y):  
    n=len(Y)  
    es=np.zeros(n)  
    for i in range(n):  
        p=P[i,:]  
        y=Y[i]  
        es[i]=entropia_ejemplo(p, y)  
    return es.mean()
```

Promedio = 1.41

Keras - Con Codificación por Etiquetas

```
x,y=cargar_dataset(one_hot=False)
nx,d_in  = x.shape # x tiene tamaño n x d_in
ny = y.shape # y tiene tamaño n x 1 (codificación etiquetas)
classes=y.max()+1 # etiqueta de clase máxima+1
assert(nx=ny) # misma cantidad de ejemplos en ambos vectores
import keras
#Creamos el modelo
model=keras.Sequential()
#Capa lineal que convierte vector de d_in a d_out dims
model.add(keras.Dense(classes,input_shape=[d_in],
                      activation="softmax"))
model.compile(loss='sparse_categorical_crossentropy',# ent. cruz.
              optimizer='sgd') # descenso de gradiente
history = model.fit(x,y,epochs=100,batch_size=32)
y_predicted=model.predict(x)
```

Keras - Con Codificación One Hot

```
x,y=cargar_dataset(one_hot=True)
nx,d_in = x.shape # x tiene tamaño n x d_in
ny,classes = y.shape # y tiene tamaño n x clases (one hot)
assert(nx=ny) # misma cantidad de ejemplos en ambos vectores

import keras
#Creamos el modelo
model=keras.Sequential()
#Capa lineal que convierte vector de d_in a d_out dims
model.add(keras.Dense(classes,input_shape=[d_in],
                      activation="softmax"))
model.compile(loss='categorical_crossentropy',# ent. cruz.
              optimizer='sgd') # descenso de gradiente
history = model.fit(x,y,epochs=100,batch_size=32)
y_predicted=model.predict(x)
```

Accuracy con varias clases

- Accuracy = % de ejemplos que clasificó correctamente

Carrera	P(Carrera)			Predicción	Acierto
	LS	LI	IC		
LS = 0, LI = 1, IC = 2					
0	0.01	0.45	0.55	2	0
1	0.09	0.39	0.52	2	0
1	0.09	0.39	0.52	2	0
2	0.02	0.73	0.24	1	0
0	0.36	0.36	0.27	0	1
2	0.68	0.15	0.17	0	0
1	0.55	0.27	0.18	0	0
0	0.91	0.05	0.05	0	1

- **Predicción**
 - $P(\text{Carrera}).\text{argmax}()$
 - El índice del mayor elemento
- **Acierto**
 - 1 si Carrera = Predicción
- **Accuracy**
 - Promedio de columna **Acierto**
- Ejemplo
 - $N = 8$
 - aciertos=2
 - $\text{Accuracy} = \text{aciertos}/N = 2/8 = 0.25$
 - $\text{Accuracy}(\%) = 25\%$

Resumen

- Permite realizar clasificación de **más de 2 clases**
- Función **softmax** se aplica luego de la regresión lineal
 - Genera una distribución de probabilidades
- **Entropía cruzada:** error adecuado para la **softmax**
- Sigue teniendo una solución única (optimización convexa)

Variables			Carrera
Estudio	Edad	Promedio	LS = 0, LI = 1, IC = 2
2	24	4	0
5	22	3	1
7	25	4	1
9	20	7	2
10	19	4	0
11	20	3	2
13,4	21	5	1
14	20	3	0