

Introducción a la Inteligencia Artificial
Facultad de Ingeniería
Universidad de Buenos Aires



Clase 5

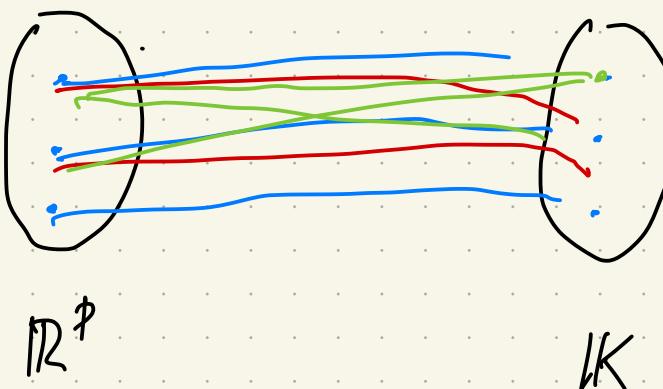
1. Clasificación Binaria
 - a. Motivación
 - b. Regresión Logística - Ejercicio de Aplicación
 - c. Regresión Logística - Teoría
2. Clasificación Multiclas
3. Ejercicio integrador



hasta ahora nosotros teníamos $x \in \mathbb{R}^{n \times p}$ e $y \in \mathbb{R}^n$
 $y \notin \mathbb{R}$ $y \in \left\{ \underbrace{\mathbb{N} \{ 1, 2, \dots \}}_{\text{Espacio ordenado}}, \mathbb{K} \{ \text{true, false} \} \right\}$

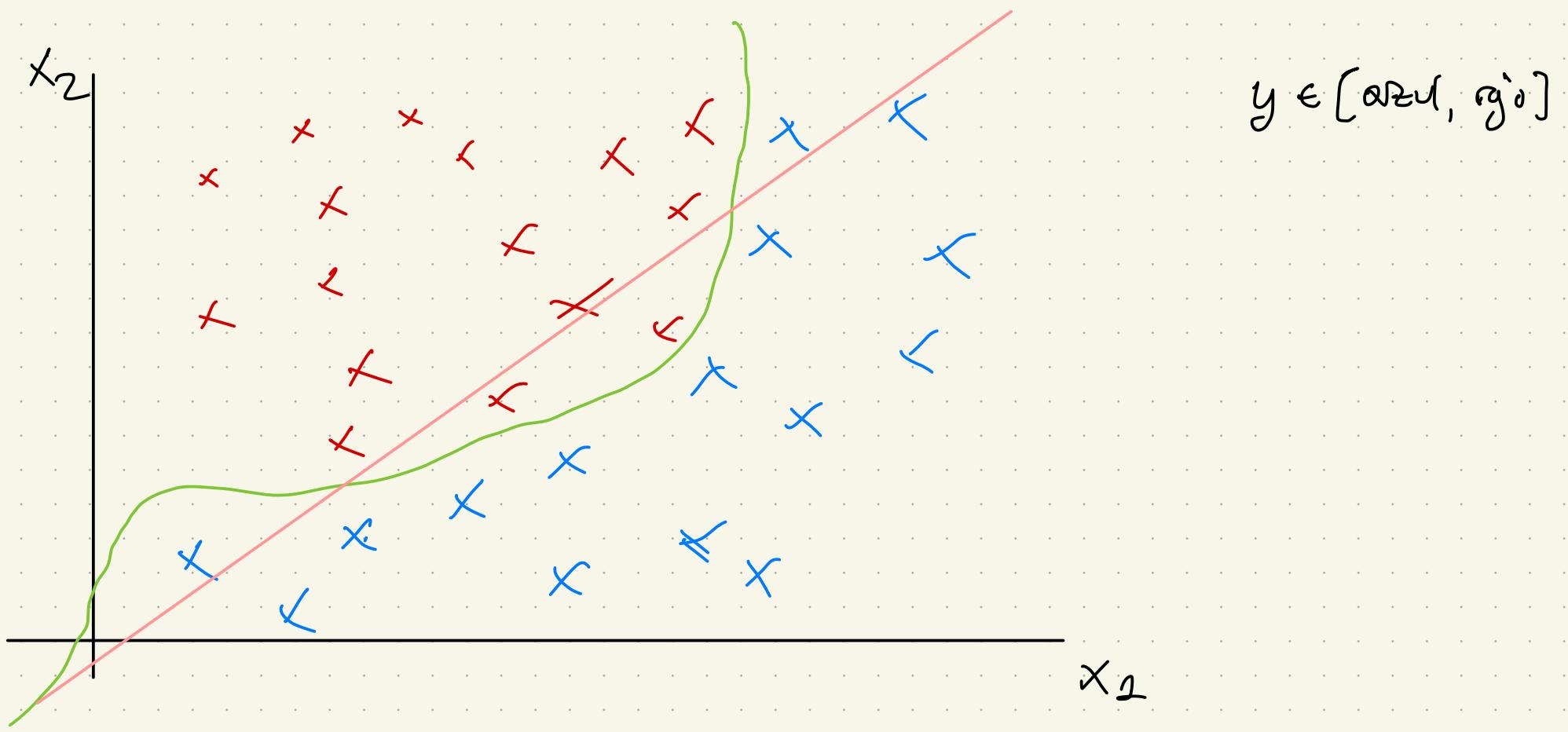
Vamos a concentrarnos en $y \in \mathbb{K}$, esto quiere decir que podemos modelar y como **función etiquetadora**:

$$f: \mathbb{R}^P \mapsto \{ 0, 1, 2, \dots, K \}$$



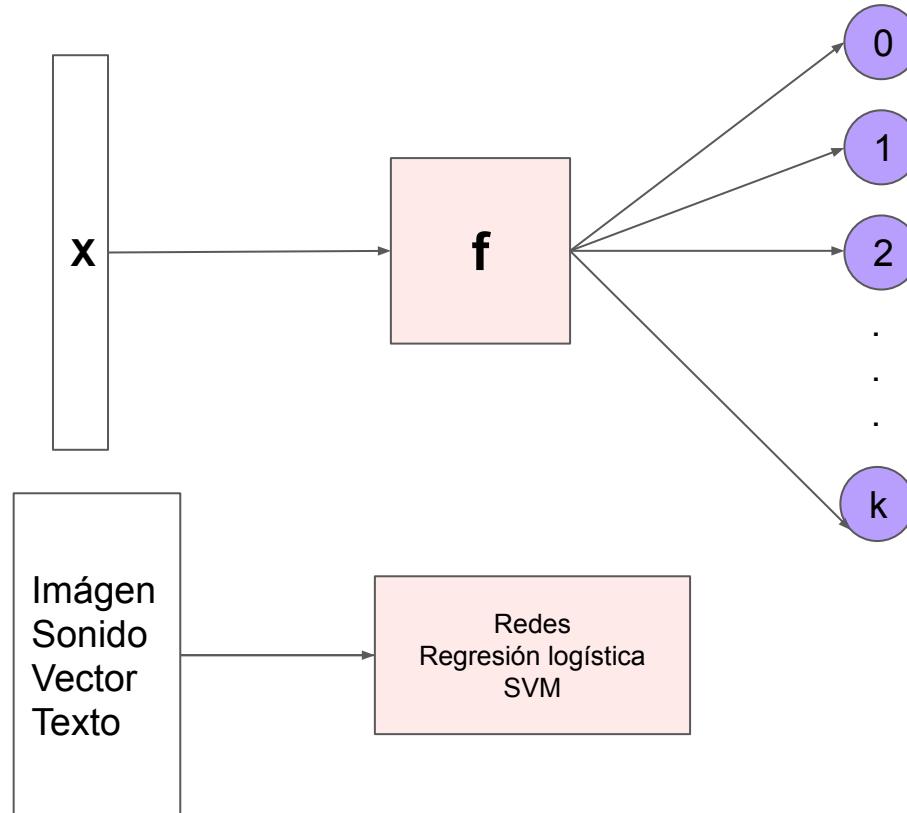
nosotros vamos a quererlos con un subconjunto de fn. f. que son **separables** (suposición super importante):

$$\forall \bar{x}_i \exists y_i | y_i \in \mathbb{K} \wedge f(y_i) \exists!$$



Esta suposición nos dice que existe una región (en port. on hiperplanos) que separa los datos.

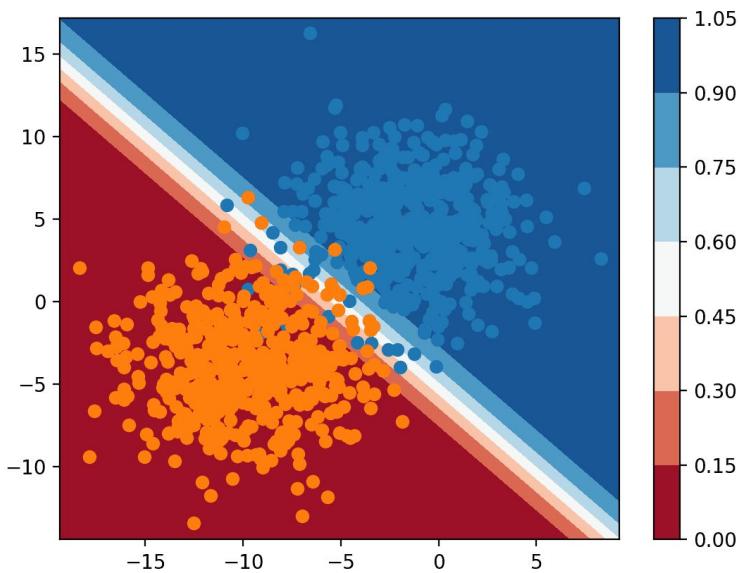
Clasificación



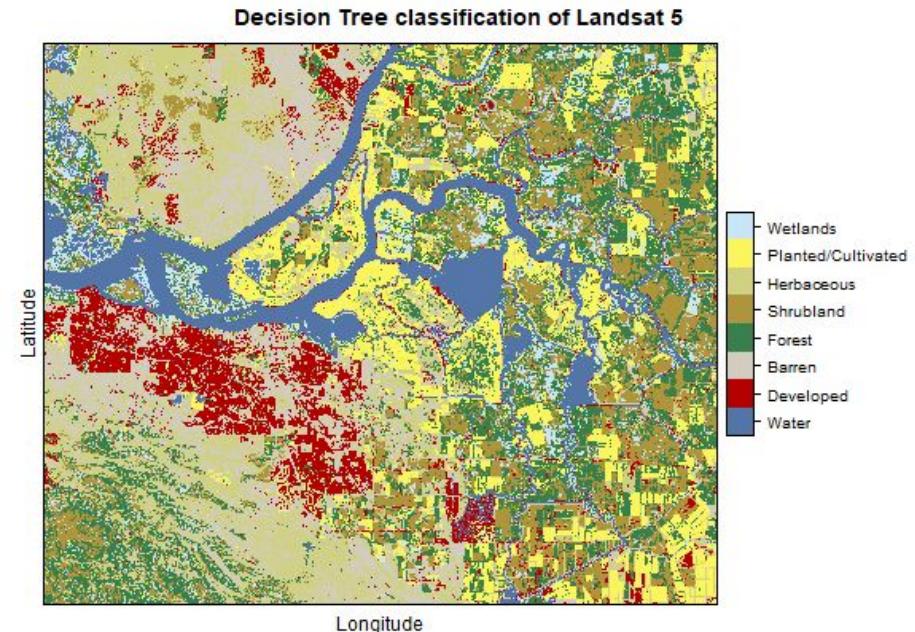
$$f : R^D \rightarrow \{0, 1, 2, \dots, k\}$$

K es un valor conocido
y fijo

Clasificación



ej. clasificación
binaria



clasif
multiclas

Clasificación binaria

. $f \rightarrow \{0,1\}$

*Suposición →
SUPER importante*

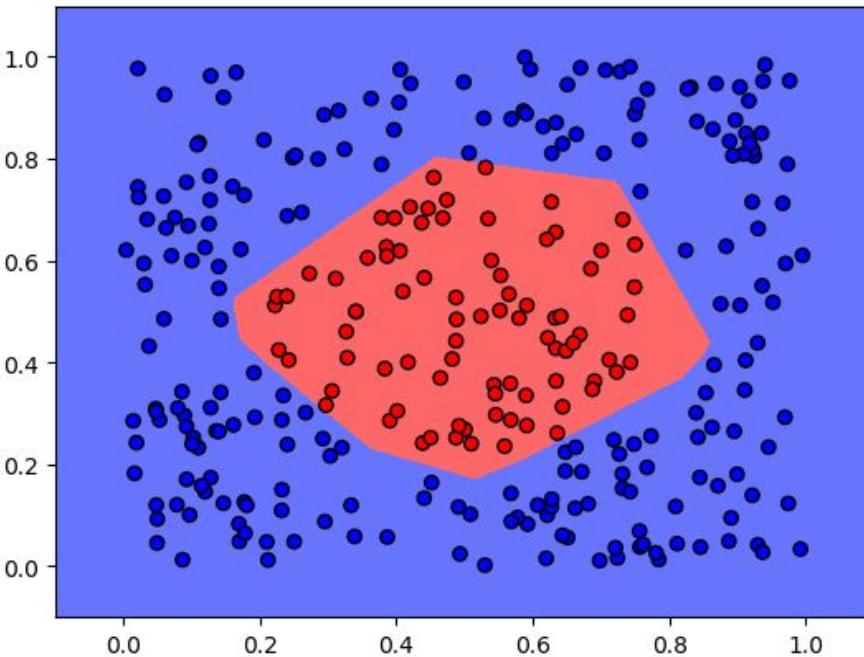
$$P(Y=0|X) = P(Y=1|X) = \frac{1}{2}$$

. $f(x_i) = \begin{cases} 1 & \text{Si } g_i \in k_1 \\ 0 & \text{o. w.} \end{cases}$

Clasificación Binaria - Ejemplos

¿ sos A ó \bar{A} ?

- Detección de fraudes
- Diagnóstico médico
- Detección de spam
- Sentiment Analysis ①
- Detección de objetos → *está* / *no está*
- Outliers ↙ *es o no outlier*



① en texto suele ser binario (+, -)

en audio suele ser multiclase (enojado, triste, ...)

Clasificación Binaria - Diagnóstico médico

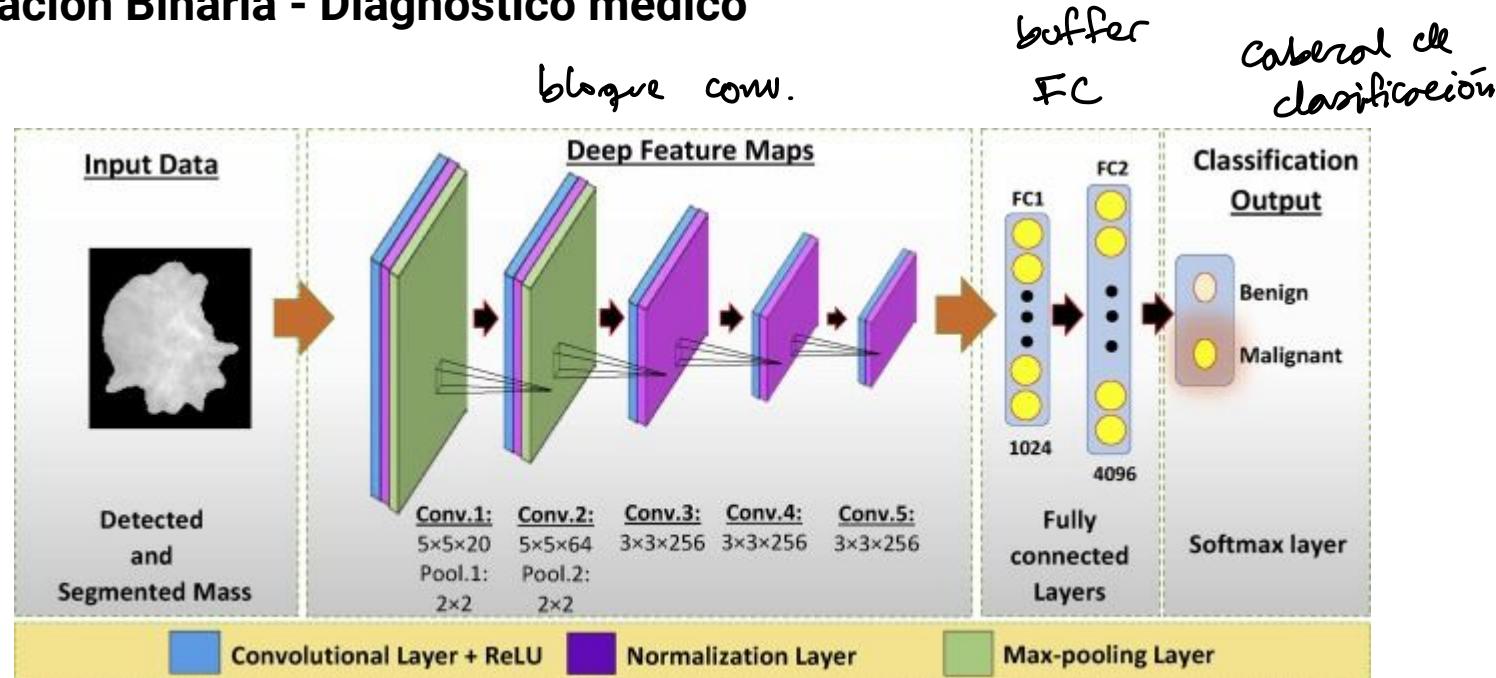


Imagen de: "A fully integrated computer-aided diagnosis system for digital X-ray mammograms via deep learning detection, segmentation, and classification"

Clasificación Binaria - Spam detection

"Spam or Jam"

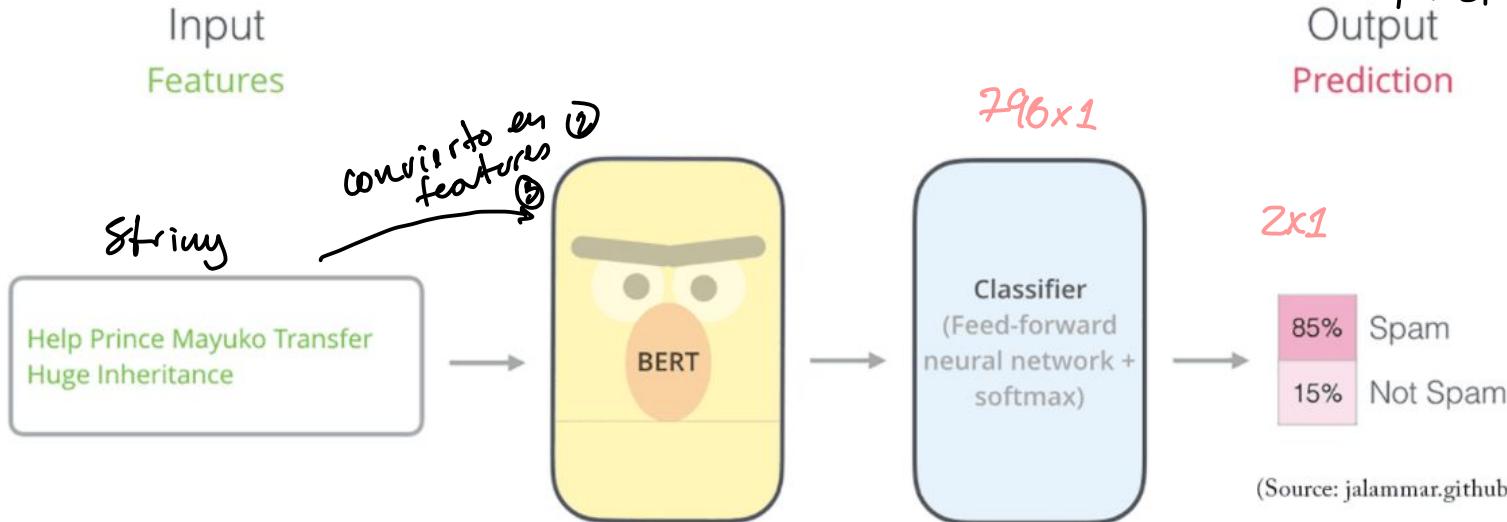


Imagen de: "Tutorial: Fine tuning BERT for Sentiment Analysis"

② Bag of Words (Bow), TF-IDF (métodos clásicos basados en freq.)

Word 2 Vec, (embeddings), UBA fiuba
Fasttext, ...

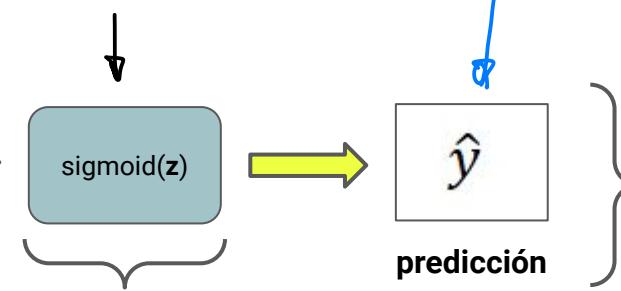
Regresión Logística

Regresión Logística

$$\begin{bmatrix} x_{i,1} \\ x_{i,2} \\ x_{i,3} \\ \vdots \\ x_{i,m} \end{bmatrix} \xrightarrow{z \in \mathbb{R}} z = b + \sum_j w_j \cdot x_j$$

x muestra i
(m features)

$$f: \mathbb{R} \mapsto [0,1]$$



\hat{y} puede ser
o label probabilidad

¿Función de Costo?
 $\mathcal{L}(y, \hat{y})$

$\mathcal{L}(\text{gato}, \text{perro})$

Métricas: en clasif. no podemos usar lo que veniamos usando

MSE (gato, perro) ? \rightarrow en clasif. usamos la matriz de confusión.

Pred	T	F
real \ Pred	TP	FN
GL*	+	
F	FP	TN

\leftarrow prediction

$$y = \mathbb{I}_{c=k} = \begin{cases} 1 & \text{si } y_i = k \\ 0 & \text{si } y_i \neq k \end{cases}$$

• Accuracy = $\frac{TP + TN}{\# \text{muestas}} \in [0,1]$

• Specificity = $\frac{TN}{TN + FP}$
(TN rate)

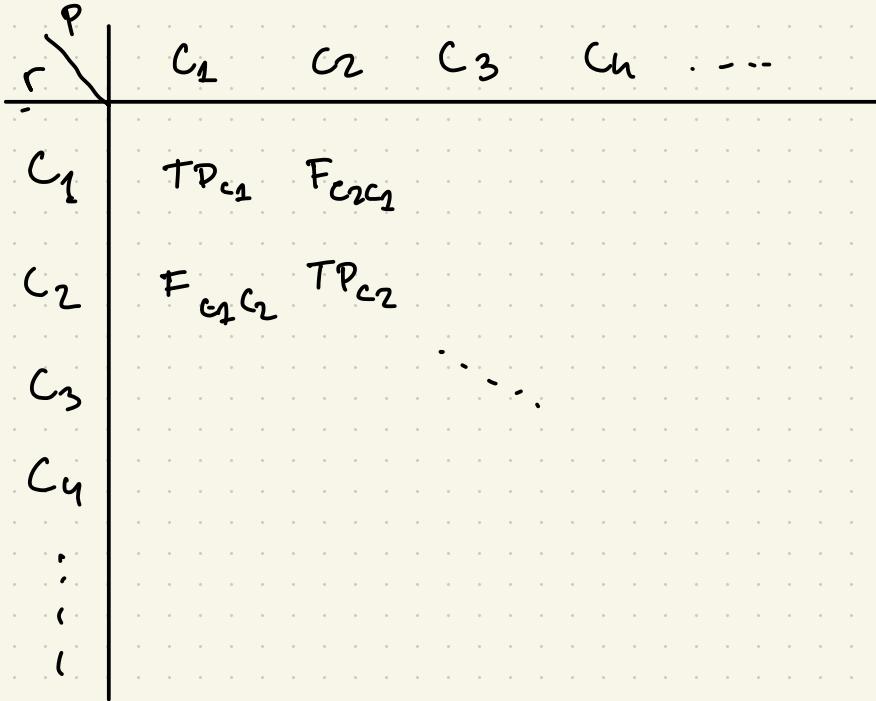
• precision = $\frac{TP}{TP + FP}$

• recall = $\frac{TP}{TP + FN}$

• $F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$

• $F_{\beta} = (1+\beta)^2 \frac{\text{prec. recall}}{\beta^2 \text{prec} + \text{recall}}$ ($F_{0,25}$)

* GL: Golden Label
Ground truth
Real values



Δc_{c_1}

Δc_{c_2}

\vdots
 \vdots

Macro Avg : promedio ponderado

Micro Avg : promedio

Formas de clasificar:

+ Definimos un proc. de scoring \rightarrow como resolvemos una clasif.

↑ complejidad \Rightarrow ↑ computo \Rightarrow mejor modelo?

1. **Modelos generativos**: modelar las distib. de I/O ms generar la salida (permite generar información sintética) \rightarrow GAN's

2. **Modelos discriminantes**: plantear $P(C_k | \bar{x})$ ms utilizar métodos de inferencia (bayesiana) para estimar P . Seteamos $P(C_k)$ como distrib. a priori.

3. **modelos de función discriminadora**: buscamos $f: \mathbb{R}^{n \times m} \mapsto \{C_1, \dots, C_K\}$

↳ Regresión logística (la usamos porque queremos explotar lo conocido).

partimos de suponer $y \sim \text{Bernoulli}(1, \pi_i) \rightarrow$

$$\begin{cases} E(y) = \pi_i \\ \text{Var}(y) = \pi_i \cdot (1 - \pi_i) \end{cases}$$

me gustaría poder obtener algo así:

$$\pi_i \sim \bar{x}^t \cdot \bar{\beta} \quad ; \quad \bar{x} \in \mathbb{R}^{n \times p}, \bar{\beta} \in \mathbb{R}^{p \times 1}$$

$\pi_i \in [0,1]$, pero $\bar{x}^t \cdot \bar{\beta} \in \mathbb{R} \Rightarrow$ tengo que plantear una fn.
de mapeo.

buscamos $f: [0,1] \mapsto \mathbb{R}_{\geq 0}$ and $\text{odds}_i = \frac{\pi_i}{1-\pi_i}$ razón de probabilidad

$$\text{odds}_i = \frac{P(A)}{P(\bar{A})}$$

Vamos a tomar el log de odds, esto me permite definir una función biyectiva $\mathbb{R}_{\geq 0} \mapsto \mathbb{R}$. Esta transf. se llama **logit**
(log odds ratio):

$$\eta_i = \text{logit}(\pi_i) = \bar{x}^t \cdot \bar{\beta}$$

existe antilogit: $\pi_i = \text{antilogit}(\eta_i) =$

$$\frac{e^{\eta_i}}{1 + e^{\eta_i}}$$

fn. sigmoidal
fn. logística

Regresión Logística

Logistic function

→ es una fn. de la familia de *Squashing*.

$$S(x) = \sigma(x) \in [0,1]$$

$$\sigma(x)$$

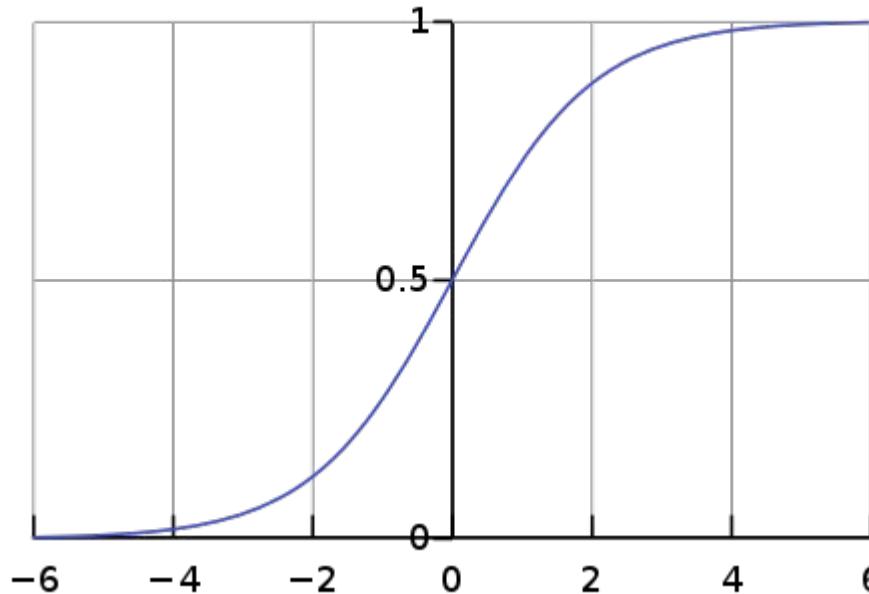
$$\sigma(x)$$

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} = 1 - S(-x).$$

$$S, \sigma: \mathbb{R} \mapsto [0,1]$$

$$\sigma(ax)$$

$$\sigma(x, a)$$



Modelo de Regresión Logística:

Sean y_1, \dots, y_n realizaciones de un proc Bernoulli ($1, \pi_i$), asumimos que existe una relación **lineal** entre x_i (datos) y el **logit** de la razón de prob. de π_i :

$$\text{logit}(\pi_i) = b + \sum_{j=0}^p w_j \cdot x_{ij}$$

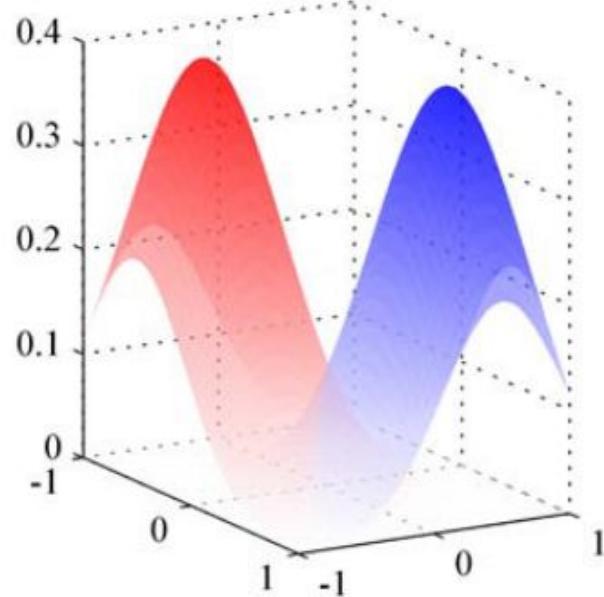
Este modelo es parte de la familia de modelos lineales generativos
→ Es equivalente a decir modelo lineal generalizado con respuesta binomial y fn. de enlace logit.

$$g(E(Y|X)) = b + \sum_i w_i x_i$$

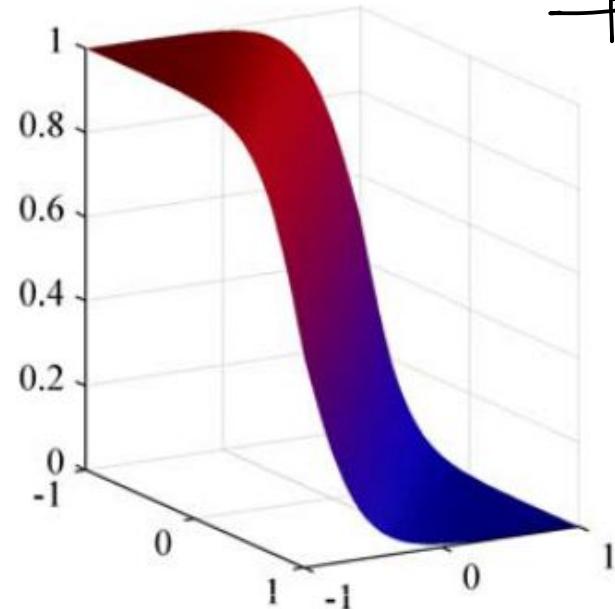
$Y \sim f$; f distnb. de la respuesta $\sim g$ es la función de link

Regresión Logística

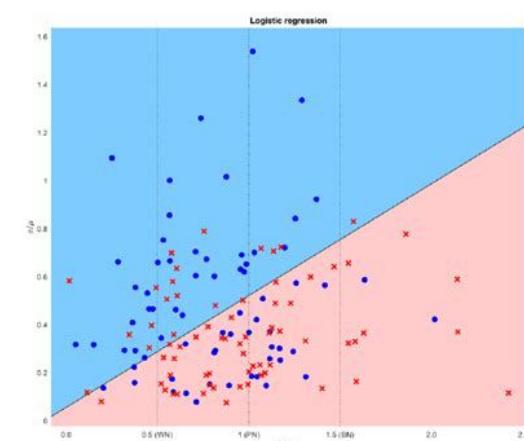
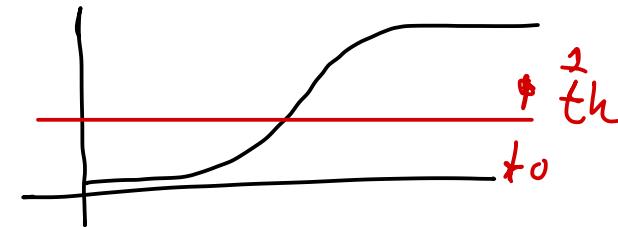
Regresión Logística



Class-conditional - $P(x|C_n)$



Posterior - $P(C_n|x)$



¿Cómo obtenemos w ?

queremos mapear $\sigma(z) = \sigma(w^t x)$

$$\partial_w \sigma(z) = \sigma(w^t x) (1 - \sigma(w^t x)) \quad \textcircled{1}$$

planteamos la fn. de verosimilitud: $P(y/x) = \prod_{i=1}^N \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i} = L$

{ N: cant. de muestras

y: goleto label

\hat{y} : probabilidad de ser una clase

$$l = \sum_{i=1}^N \ln(P_w(y=y_i | x=x_i)) \Rightarrow \text{buscamos optimizar } l$$

$$\max_w \sum_i \ln(\sigma(w^t x)^{y_i} \cdot (1 - \sigma(w^t x))^{1-y_i})$$

$$\max_w \sum_i y_i \underbrace{\ln \sigma(w^t x)}_{\textcircled{A}} + (1-y_i) \underbrace{\ln (1 - \sigma(w^t x))}_{\textcircled{B}}$$

Propiedad copula:

$$\partial_a \sigma(a) = \sigma(a)(1-\sigma(a))$$

multiplicar x (-1) :

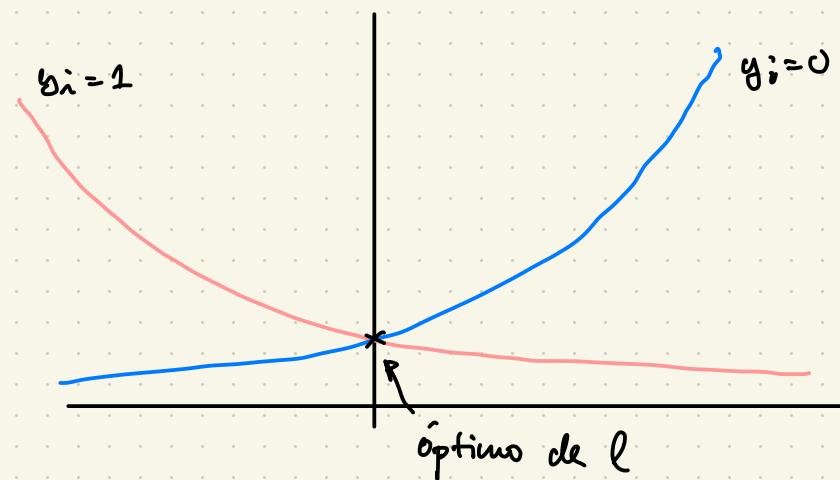
$$\min_w \sum_i -y_i \underset{P(x)}{\textcircled{A}} -(1-y_i) \underset{P(\bar{x})}{\textcircled{B}}$$

mo si minimizo el encontro el óptimo

podemos buscar el mínimo usando la crossentropy) :

$$J(w) = \frac{1}{N} \sum_i y_i \ln \sigma(wx) - (1-y_i) \ln(1-\sigma(wx))$$

la entropia da un valor real
entropia binaria cruzada (binary crossentropy)

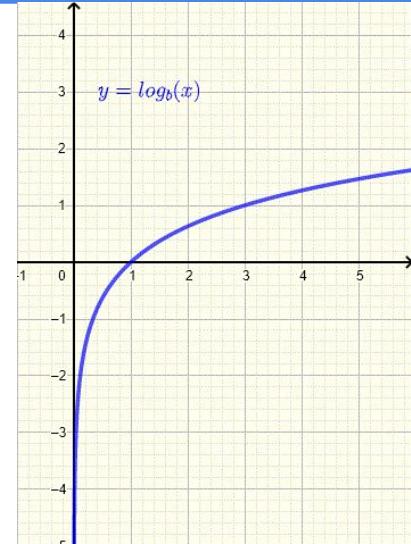
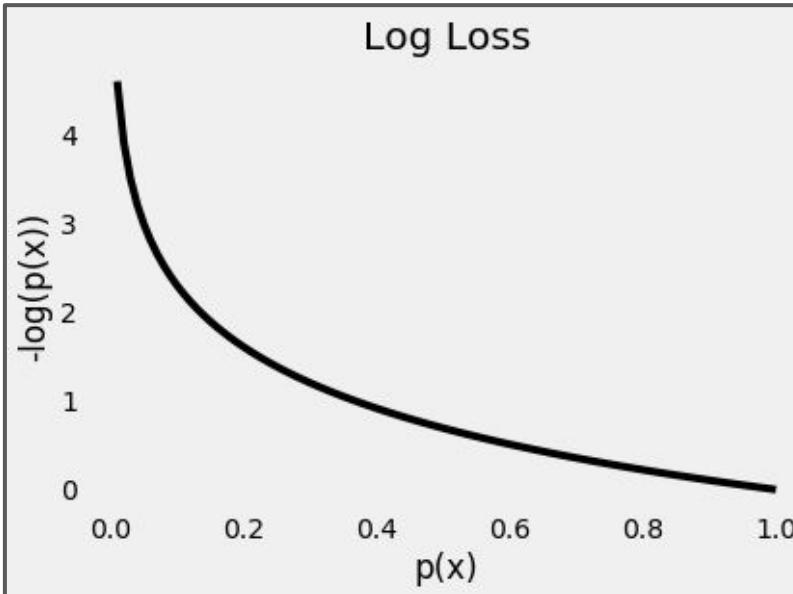


$$\nabla J_w = 0$$

↳ GD
SGD
GD MB

Función de costo - Binary cross entropy

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$



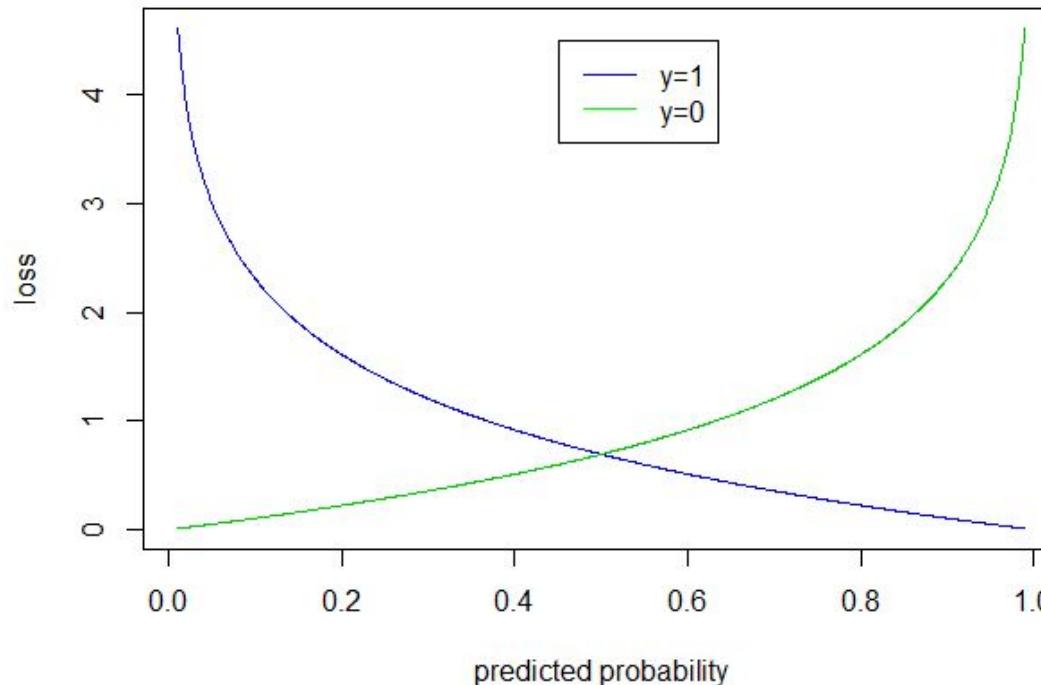
Función de costo - Binary cross entropy

$$\log \phi(y_i) = \log(\sigma(w^T x))$$

↑ ↑
dato prediccion

lo puedes calcular

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

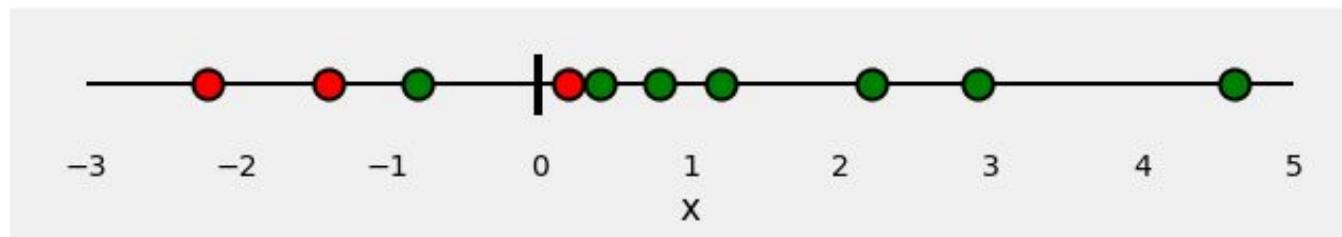
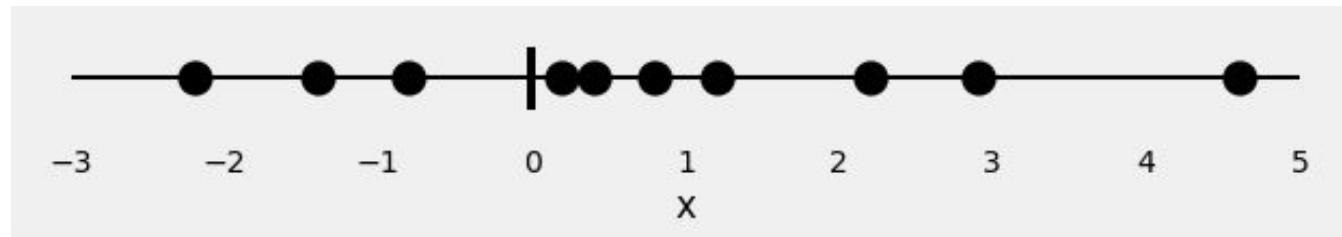


$$w_{n+1} = w_n + \alpha \nabla w$$

∇w ?

Regresión Logística

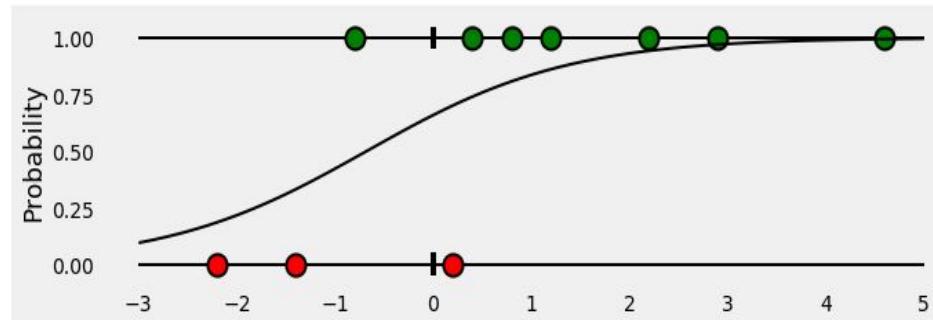
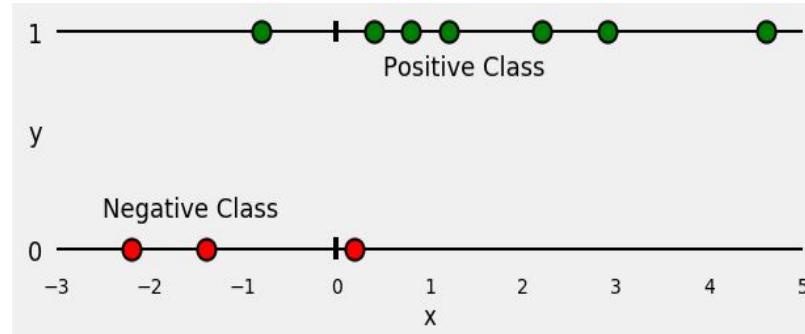
Regresión Logística $\pi_i \in [0,1] \rightsquigarrow \text{logit}(\pi_i) \in \mathbb{R}$



1: Verde, 0: Rojo

Regresión Logística

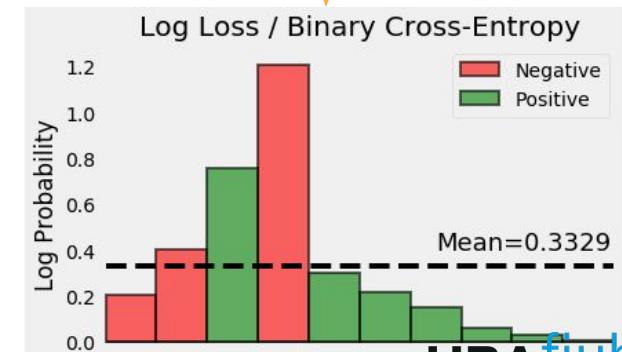
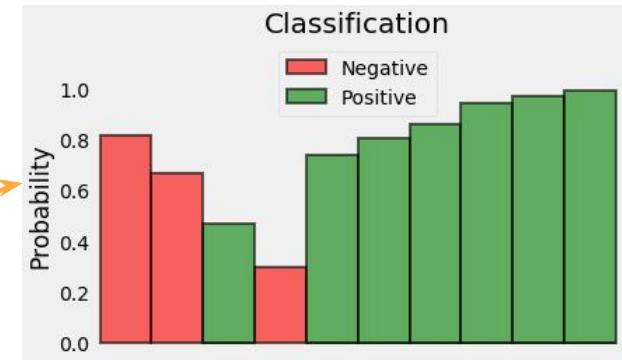
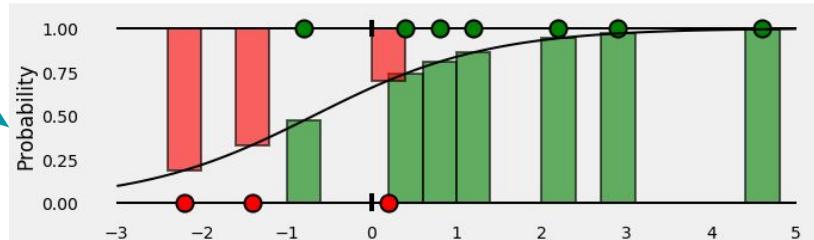
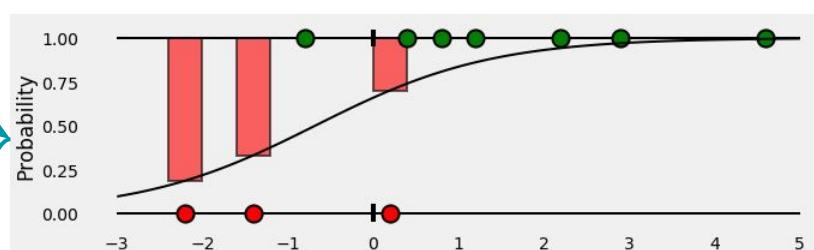
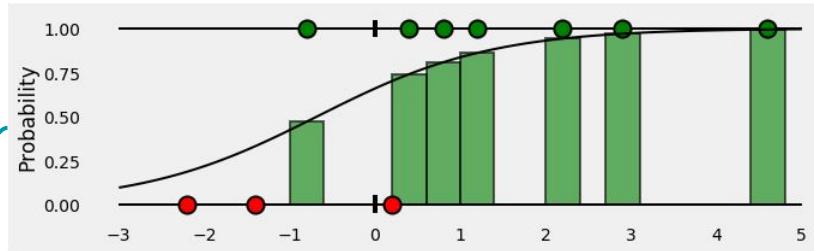
Regresión Logística

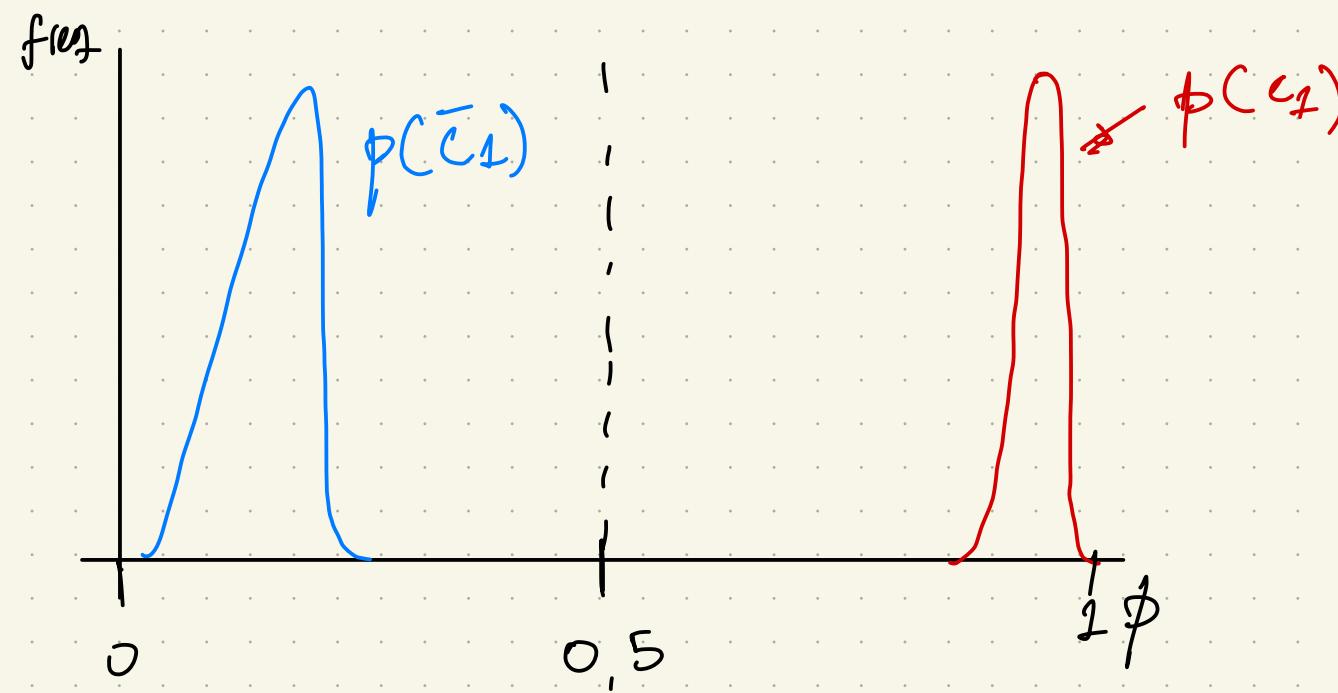


$$\hat{y} = \begin{pmatrix} \theta_0 & \theta_1 \\ \theta_2 & \theta_3 \end{pmatrix}$$

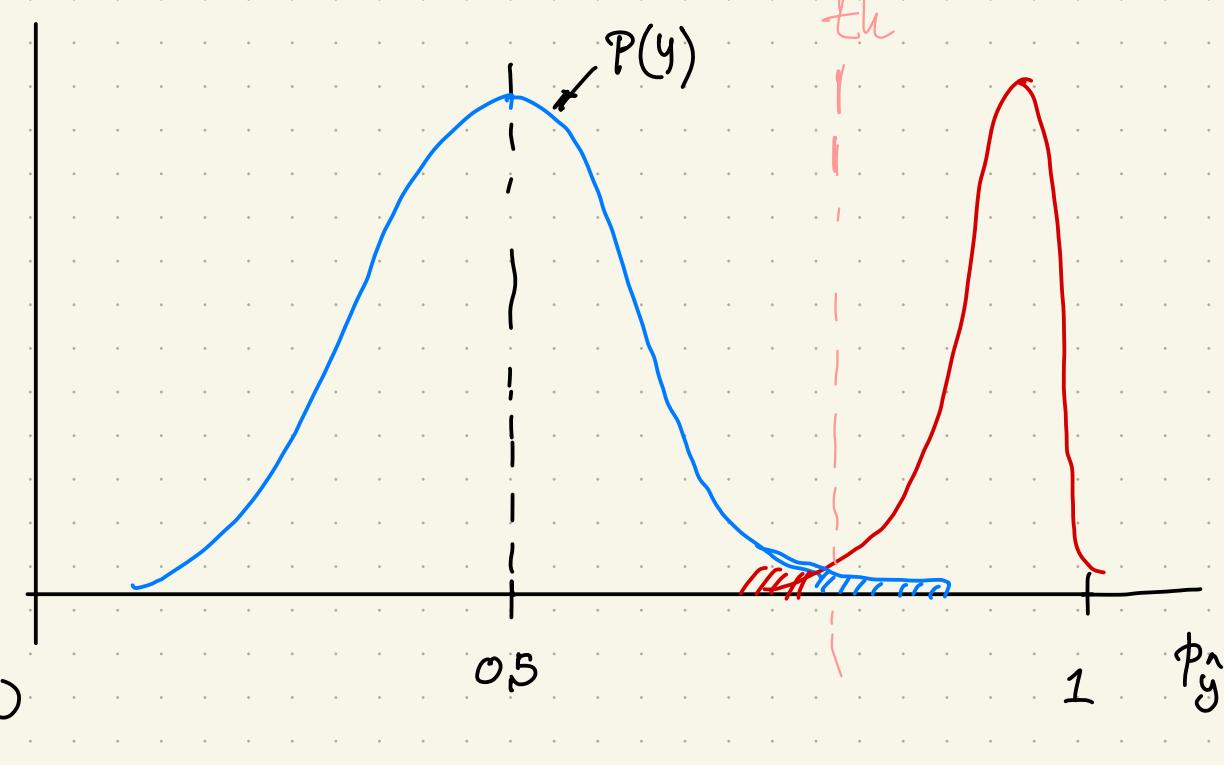
Regresión Logística

Regresión Logística





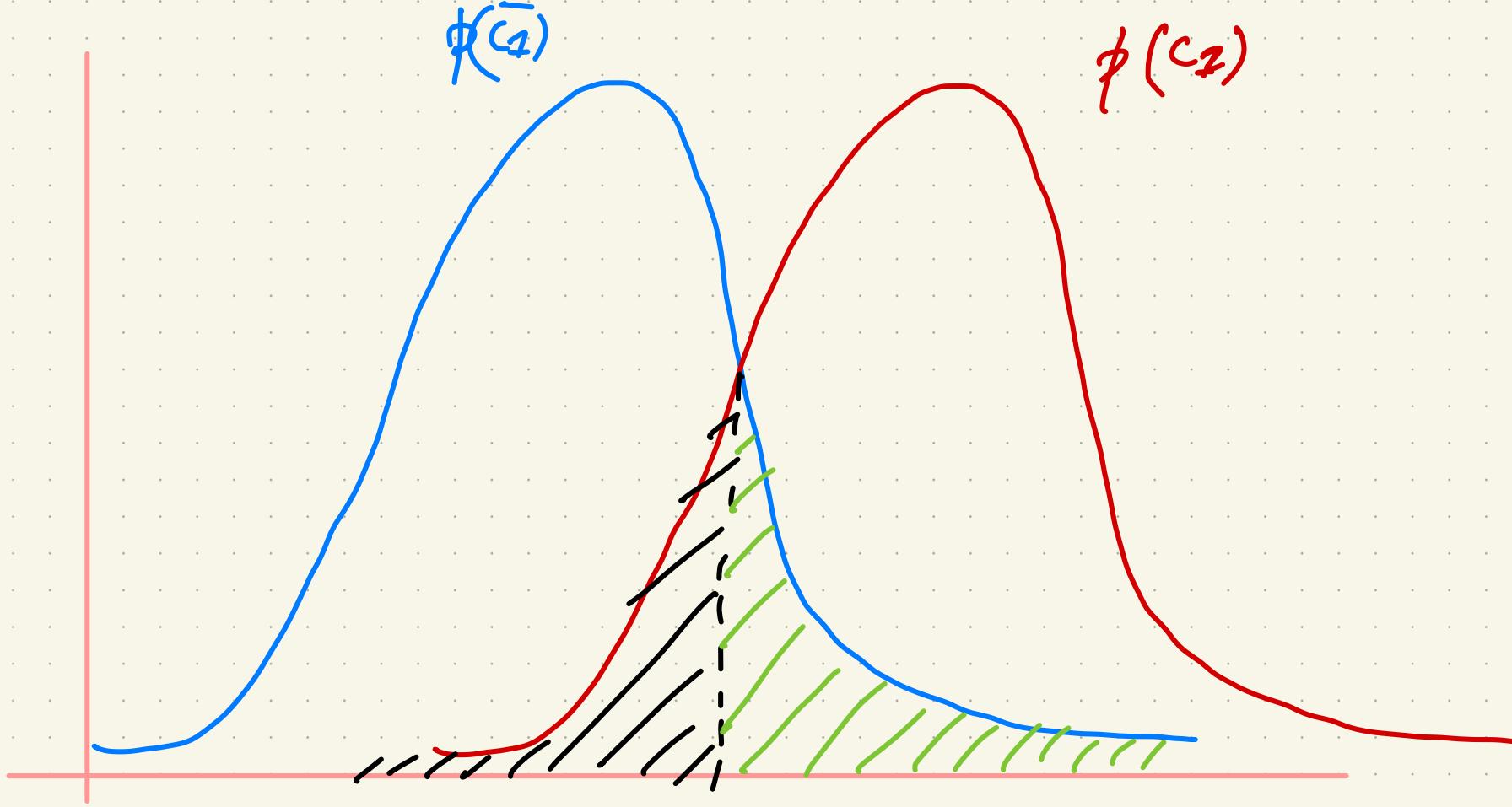
$$\hat{y} = \begin{cases} 1 & \text{si } p \geq 0.5 \\ 0 & \text{o. w} \end{cases}$$



$$\hat{y} = \begin{cases} 1 & \text{si } p > \text{th} \\ 0 & \text{o. w} \end{cases}$$

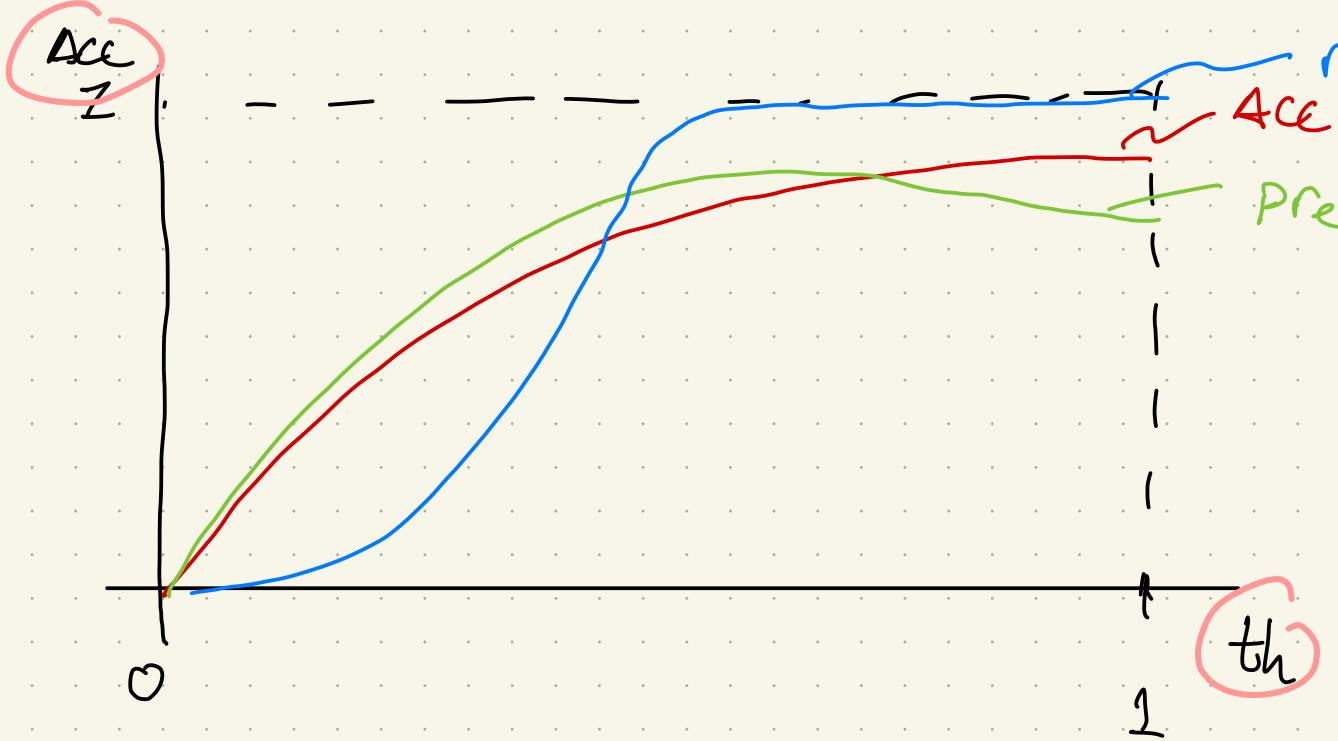
donde th lo voy a elegir
de acuerdo a mis métricas

precision (th) acc(th)
recall (th)



Aquí tenemos un problema con los errores grandes. ¿Cómo solucionamos? \rightarrow lógica de 3 estados

$$\hat{y} \in [0, 1, \text{NS/NC}]$$



buscamos un threshold que maximice por ej. F_1 . y tenemos fijado el th. calculo la matriz de confusión final.

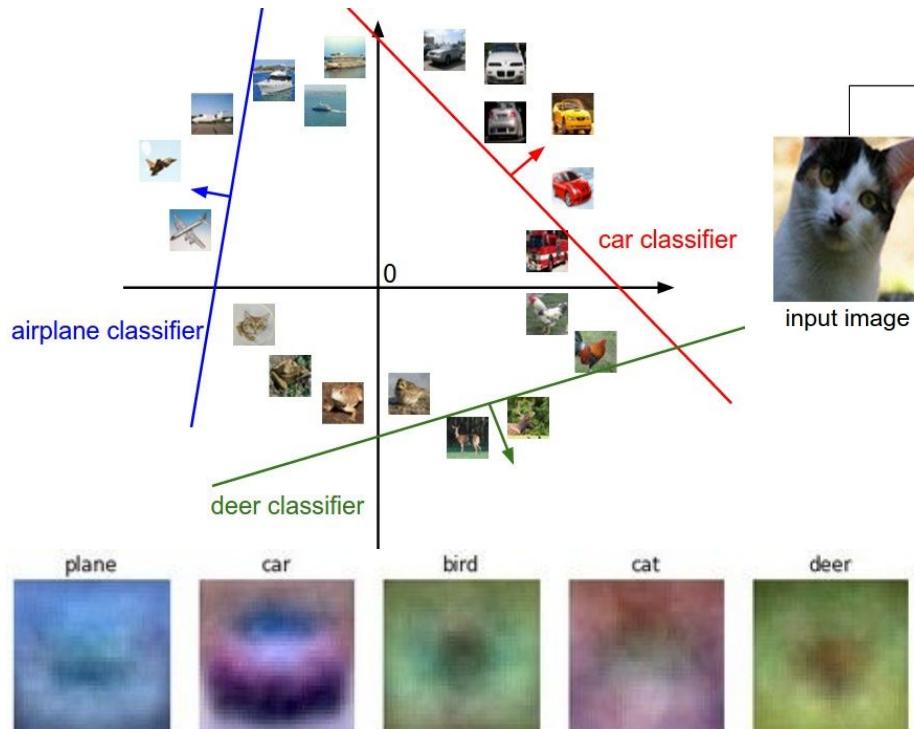
nota: cuando los prob $p(y=0)$ y $p(y=1)$ tienen mucho solapamiento. A veces conviene tener un tercer estado,

$$y \in \{ \text{True}, \text{False}, \text{NS/NC} \}$$

Clasificación multiclas

$$f = \mathbb{I}_{(y=\pi_i)}$$

Clasificación Multiclas - Motivación



stretch pixels into single column

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W

56
231
24
2

x_i

+

1.1
3.2
-1.2

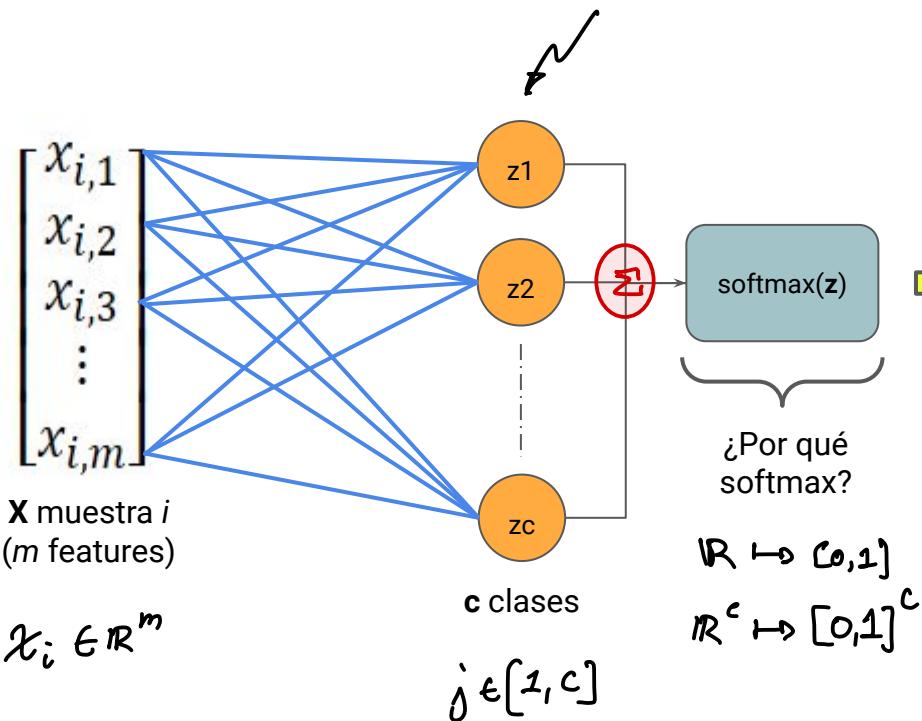
b

-96.8
437.9
61.95

$f(x_i; W, b)$

cat score
dog score
ship score

Softmax



$$\begin{bmatrix} \frac{e^{z_1}}{\sum_{j=1}^c e^{z_j}} \\ \frac{e^{z_2}}{\sum_{j=1}^c e^{z_j}} \\ \vdots \\ \frac{e^{z_c}}{\sum_{j=1}^c e^{z_j}} \end{bmatrix}$$

predicción

$$\hat{y} \rightarrow$$

$$\begin{bmatrix} 94.6 \\ 64.2 \\ \vdots \\ 8 \end{bmatrix}$$

matriz de conf. multilabel:

\backslash pred real	C_1	C_2	\dots	C_c	
C_1	TP_{C_1}	Mc_{2C_1}	\dots		Acc_{C_1}
C_2	Mc_{1C_2}	TP_{C_2}	\dots		Acc_{C_2}
:	:	:	\ddots		:
C_c	:	:		TP_{C_c}	Acc_{C_c}

los métricos ahora
son por clase:

A continuación tenemos el overall del modelo:

Acc_{mean} \rightarrow promedio

Acc_{macro} \rightarrow \cup peso do por clase

Acc_{micro} \rightarrow \cup \cup \cup soporte

Softmax

j : es la clase

C clases $\Rightarrow j \in \{1, \dots, C\}$

$$\text{odds} = \frac{P(A)}{P(\bar{A})}$$

$$P(y_i | x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

$$\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} = \frac{Ce^{f_{y_i}}}{C \sum_j e^{f_j}} = \frac{e^{f_{y_i} + \log C}}{\sum_j e^{f_j + \log C}}$$

$q(x)$

$$H(p, q) = - \sum_x p(x) \log q(x)$$

1

[Softmax Forma Gráfica](#)

2

[Softmax Visualización 3D](#)



Aca hay varios pasos intermedios

- Construir el estimador (ML-ish)
- Vamos a definir una fn. de perdida
- Vamos a comparar con algo q' conocemos

Softmax

Derivación Softmax

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}$$

$$\frac{\partial p_i}{\partial z_k} = \frac{\partial \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}}{\partial z_k}$$

$$\frac{\partial p_i}{\partial z_k} = p_i(\delta_{ik} - p_k) \quad \delta_{ik} = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases}$$

Usar gradiente descendente para actualizar W !!!

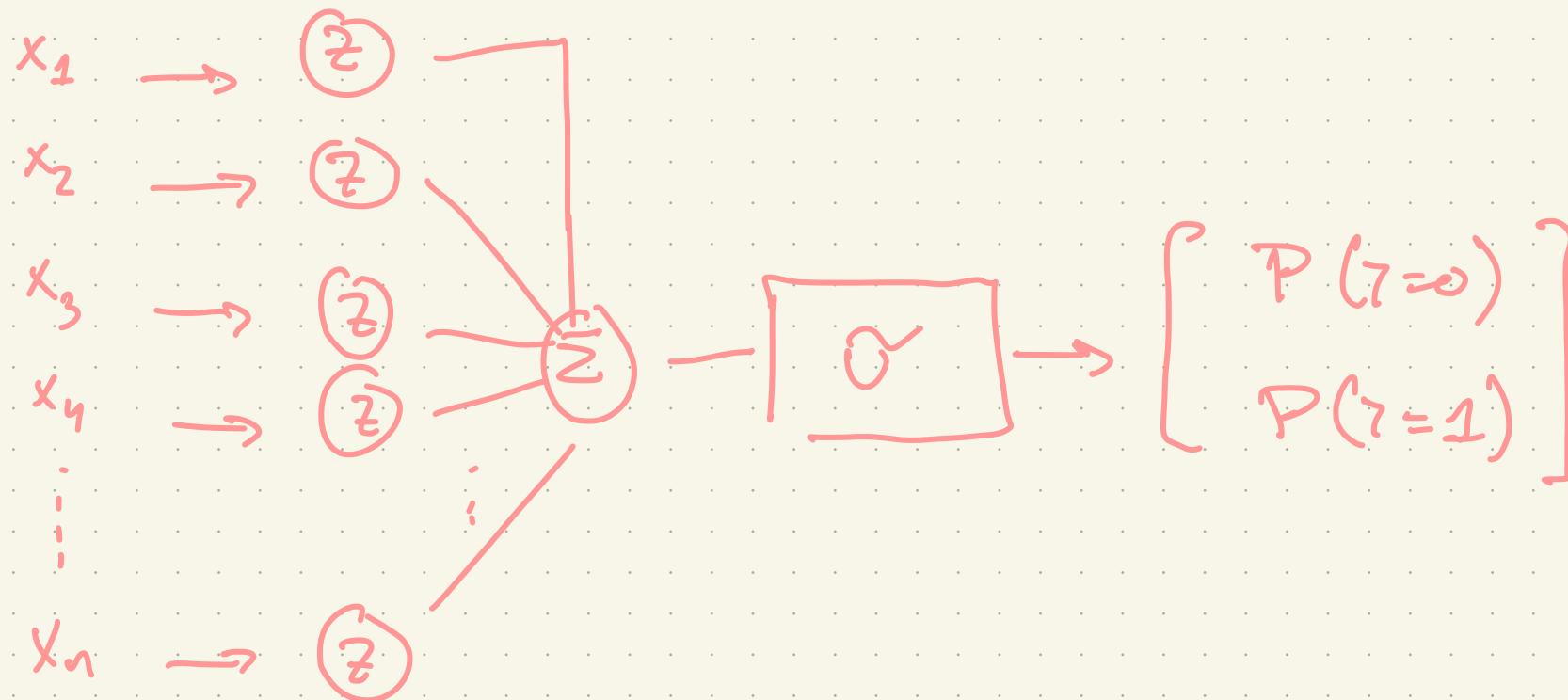
Derivación Cross-Entropy

$$\begin{aligned} L &= - \sum_i y_i \log(p_i) \\ \frac{\partial L}{\partial z_i} &= - \sum_j y_j \frac{\partial \log(p_j)}{\partial z_i} \\ &= - \sum_j y_j \frac{\partial \log(p_j)}{\partial p_j} \times \frac{\partial p_j}{\partial z_i} \\ &= - \sum_j y_j \frac{1}{p_j} \times \frac{\partial p_j}{\partial z_i} \end{aligned}$$
$$\begin{aligned} \frac{\partial L}{\partial z_i} &= -y_i(1 - p_i) - \sum_{j \neq i} y_j \frac{1}{p_j} (-p_j \cdot p_i) \\ &= -y_i(1 - p_i) + \sum_{j \neq i} y_j \cdot p_i \\ &= p_i \left(y_i + \sum_{j \neq i} y_j \right) - y_i \\ \frac{\partial L}{\partial z_i} &= p_i - y_i \\ \frac{\partial z_i}{\partial W} &= x_i \end{aligned}$$



$$\frac{\partial L}{\partial W} = \sum_{i=1}^N (p_i - y_i) x_i$$

Aproximación a extender los reales numerables.



$$\tilde{z}_i = \beta_{0i} + \sum_j \beta_{ji} x_{ij}$$

Vamos a considerar un vector de entrada \bar{x} , y vamos a construir una alternativa no lineal a \mathbb{Z} . llamaremos f a la siguiente función:

$$\bar{x} \in \mathbb{R}^{N \times 1}$$

$$f(\bar{x}) = b + \sum_{j=1}^J \beta_j h_j(\bar{x})$$

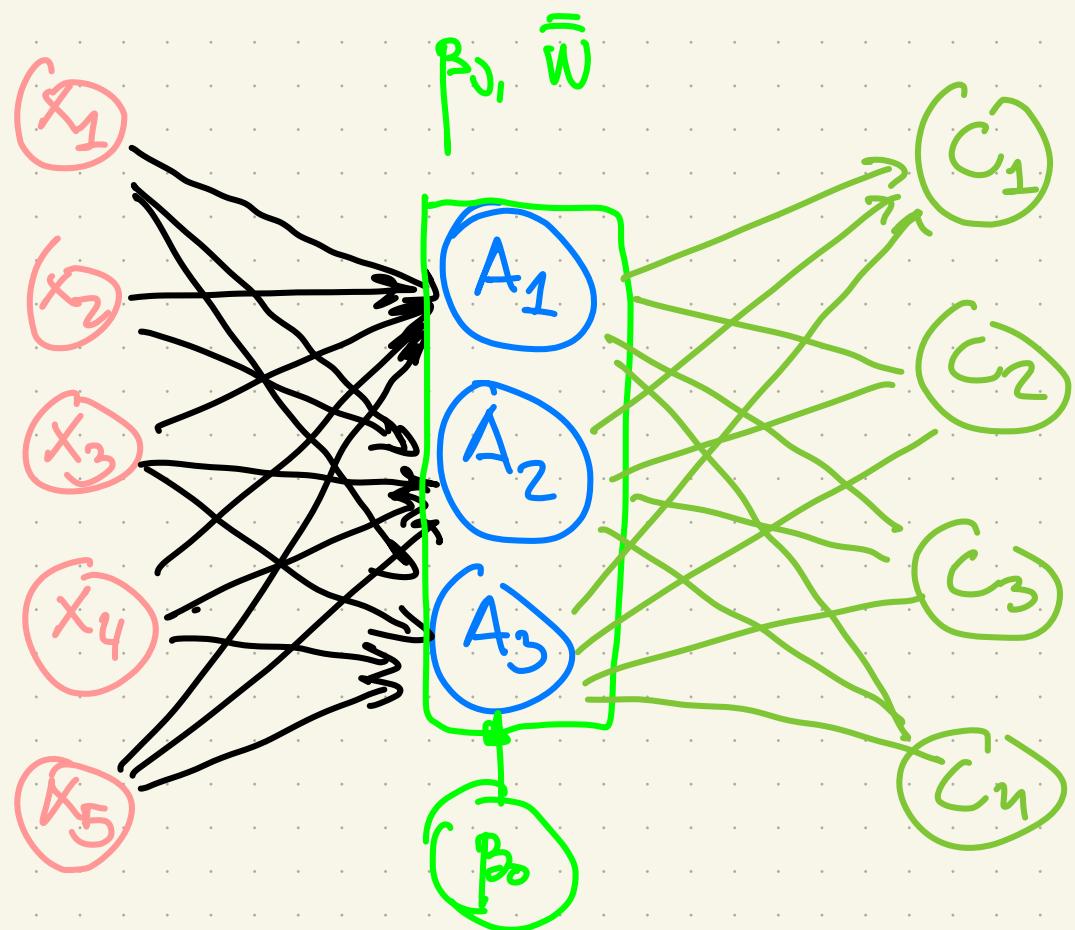
N: cant de predictores (features)

J: cant. de hidden units (neuronas p/ los amigos)

Vamos a definir h_j como $h_j = g(w_{j0} + \sum_{i=1}^N w_{ji} \cdot x_i)$

$g(z)$ se llama función de activación (es una fn. no lineal)

un ej de $g(z)$ es $\sigma(z)$ ó $\text{arctan}(z)$



1ra Capa
de la red

2da
capa

Perceptron / multilayer perceptron / feed forward \rightarrow fully connected

$$C_k = h_k(A)$$

$$= g(w_{k0} + \sum_{l=1}^L w_{kl} A_l)$$

feed forward network
fully connected

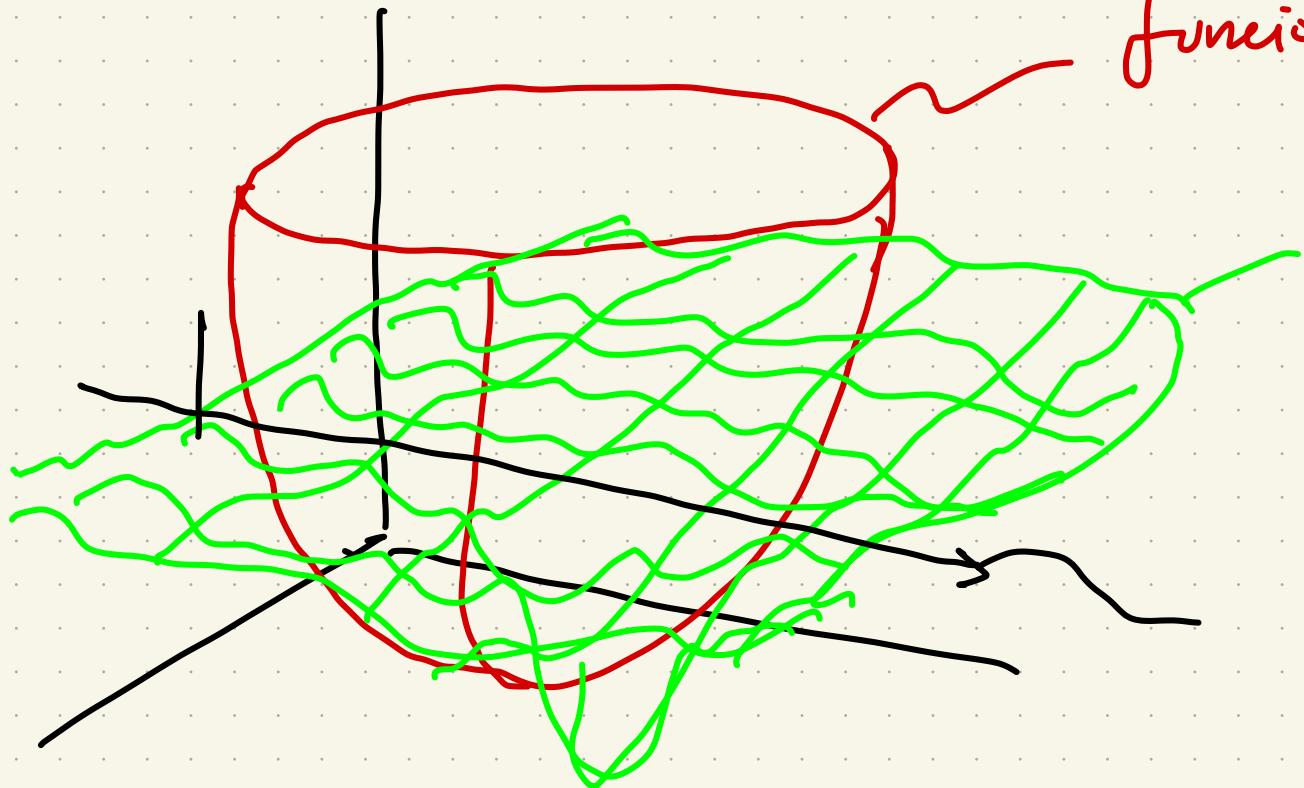
buscamos que $g(z)$ sea amigable q/ too bajar:

- $C_2 \rightarrow$ continua con deseada primera continua.
- one to one
- finito

g 's buenas: o sigmoid, arctan, una alternativa "linear"

ReLU (rectified linear unit):

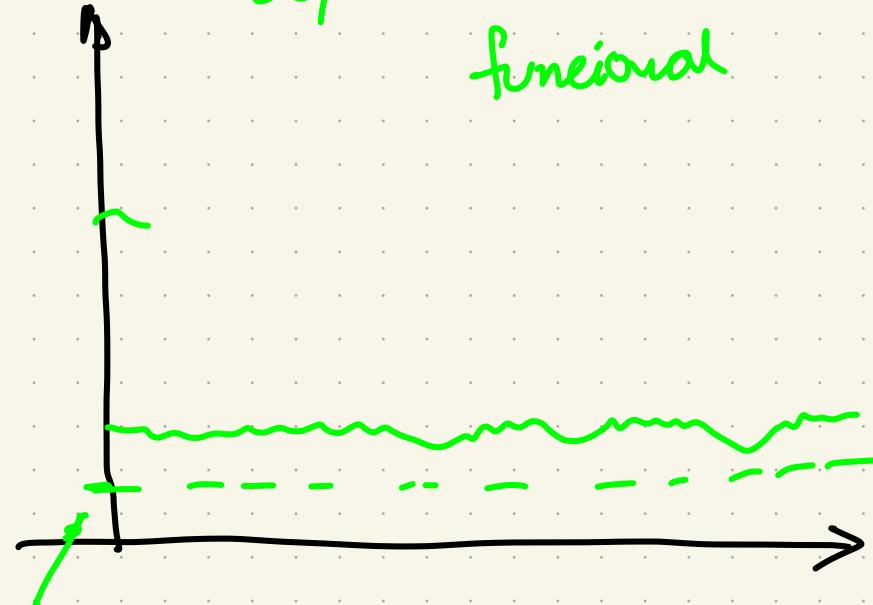
$$\text{ReLU}(z) = \begin{cases} 0 & \text{si } z < 0 \\ z & \text{o.w.} \end{cases}$$



funcional de la regresión

funcional de una RN
ANN

Supuesto corte del
funcional



óptimo más
óptimo

Bibliografía

- The Elements of Statistical Learning | Trevor Hastie | Springer
- An Introduction to Statistical Learning | Gareth James | Springer
- Deep Learning | Ian Goodfellow | <https://www.deeplearningbook.org/>
- Mathematics for Machine Learning | Deisenroth, Faisal, Ong
- Artificial Intelligence, A Modern Approach | Stuart J. Russell, Peter Norvig
- Understanding binary cross-entropy: a visual explanation | Daniel Godoy
- Visual Information Theory | [Link](#)
- <https://cs231n.github.io/>
- Classification and Loss Evaluation-Softmax and Cross Entropy Loss | Paras Dahal
- Deep learning with python - ch6 let

Lo posiblemente sea lo implemente
con pytorch