UNIVERSIDAD EAFIT

Lecture 03
Deep Learning

Ph. D. Juan David Martínez Vargas
Ph. D. Raul Andrés Castañeda

Escuela de Ciencias Aplicadas e Ingeniería

Agenda

- **Gradiente**
- **Perceptrón multicapa (MLP)**
- **Regresión Logística**
- **Sotmax**
- **Autodiferenciación (Automatic Differentiation, AD)**

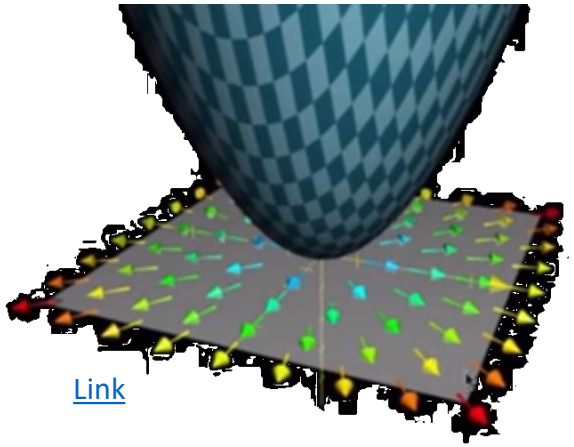
Gradiente

El operador nabla es un vector que contiene la derivada parcial a lo largo del eje unitario respectivo

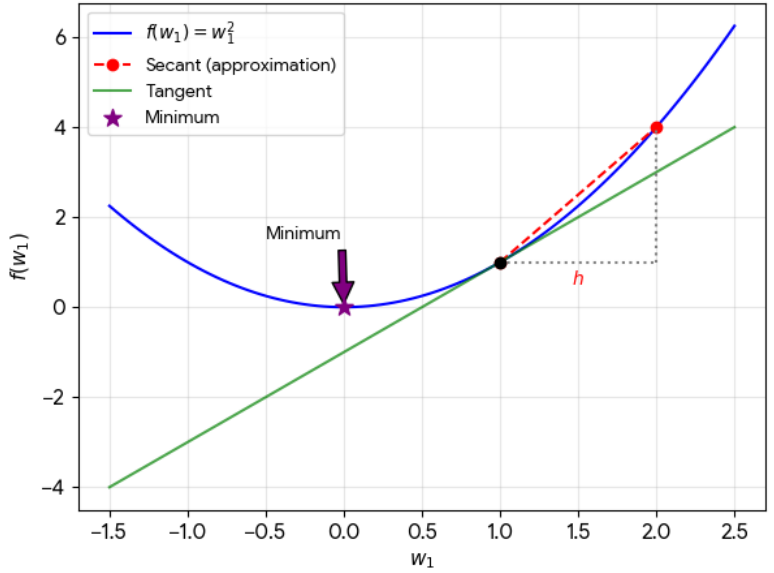
$$\vec{\nabla} \equiv \frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y}\hat{j} + \frac{\partial}{\partial z}\hat{k}$$

$$\vec{\nabla} \quad \nabla f = Gradiente \qquad \nabla \cdot f = Divergente \qquad \nabla \times f = Rotacional$$

Para un punto fijo (x_o,y_o,z_o) el gradiente le indica en qué dirección debe viajar para aumentar el valor de la función más rápidamente.



Link



∇f

Sea ***f***(x,y,z) una función escalar definida y diferenciable en cada uno de los puntos (x,y,z) de una región del espacio (*f* define un campo escalar diferenciable). El gradiente de ***f***, representado por grad(*f*), viene dado por un vector que en coordenadas cartesianas es

$$\nabla f = \frac{\partial f}{\partial x}\hat{i} + \frac{\partial f}{\partial y}\hat{j} + \frac{\partial f}{\partial z}\hat{j} \qquad \Delta f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \\ \vdots \end{bmatrix}.$$

Gradiente

Ejemplo

Para la función $f(x, y) = x^2y + y$ tendremos el gradiente $\Delta f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$,

donde

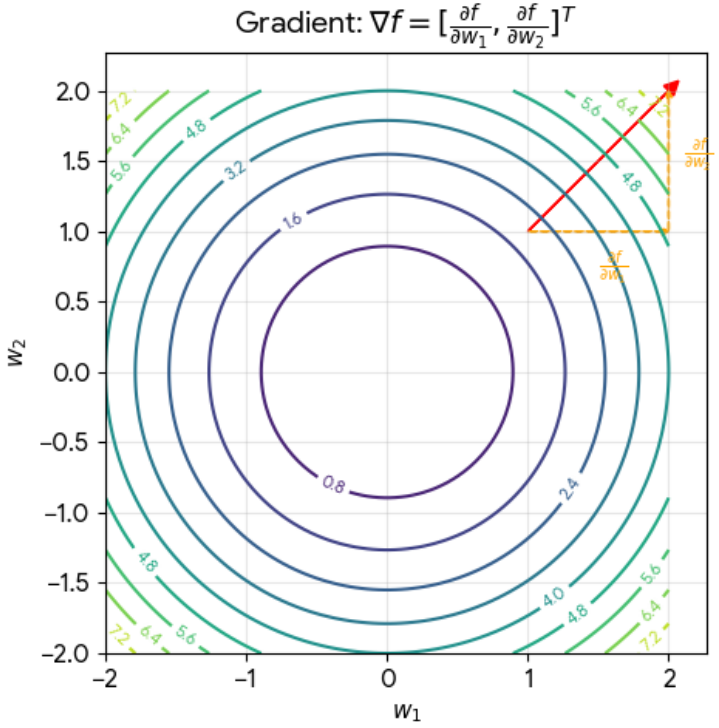
$$\frac{\partial f}{\partial x} = \frac{\partial}{\partial x} x^2y + y = 2xy$$

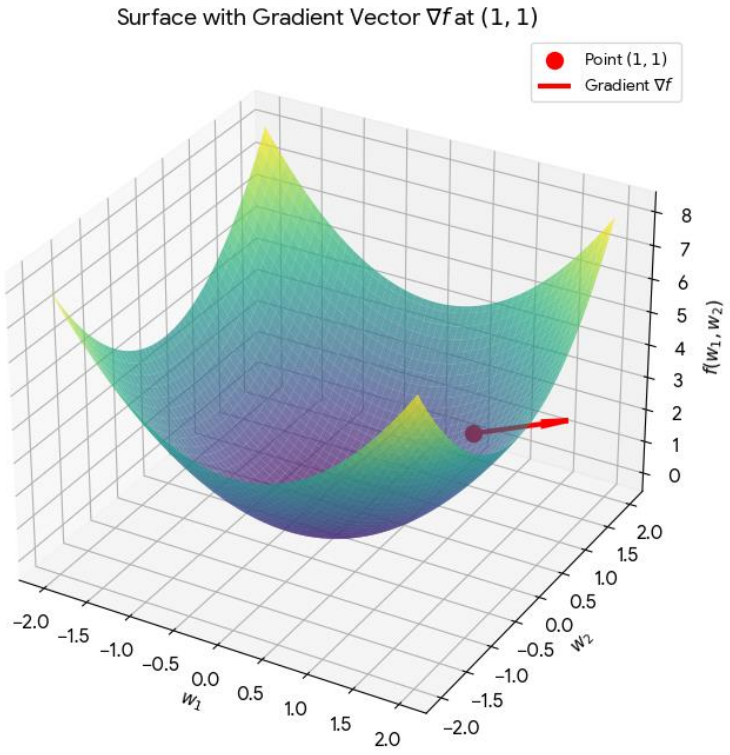
y

$$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y} x^2y + y = x^2 + 1.$$

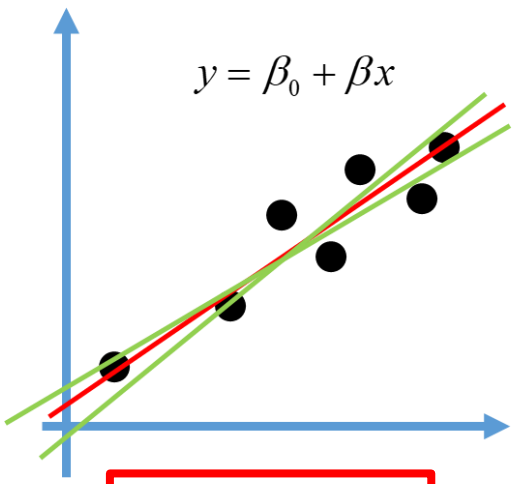
Por tanto, el gradiente de la función f es

$$\Delta f(x, y) = \begin{bmatrix} 2xy \\ x^2 + 1 \end{bmatrix}$$





Gradiente como regla de actualización

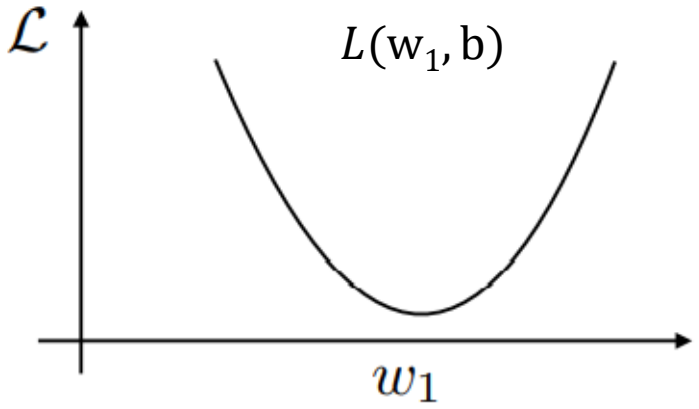


$$\beta_0 = \bar{Y} - \beta \bar{X}$$

$$\beta = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

Función de pérdida convexa

$\mathcal{L}(\mathbf{w}, b) = \sum_i (\hat{y}^{[i]} - y^{[i]})^2$



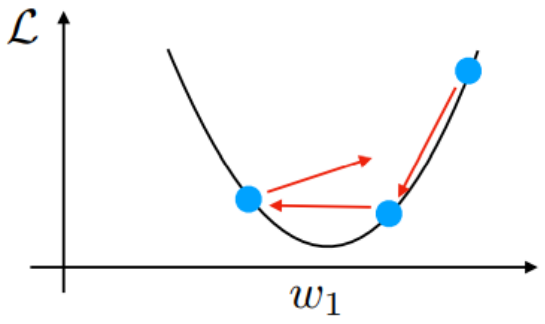
$$\beta^{(k+1)} \leftarrow \beta^{(k)} - \eta \frac{\partial J}{\partial \beta}$$

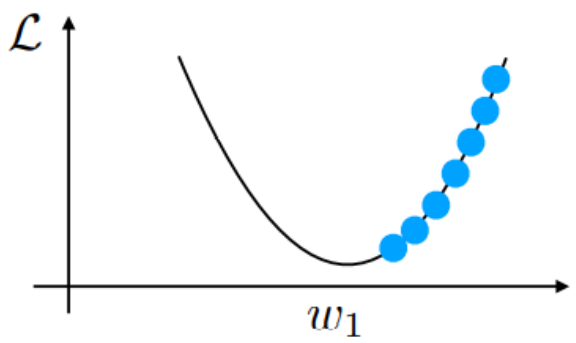
$$\beta_0^{(k+1)} \leftarrow \beta_0^{(k)} - \eta \frac{\partial J}{\partial \beta_0}$$

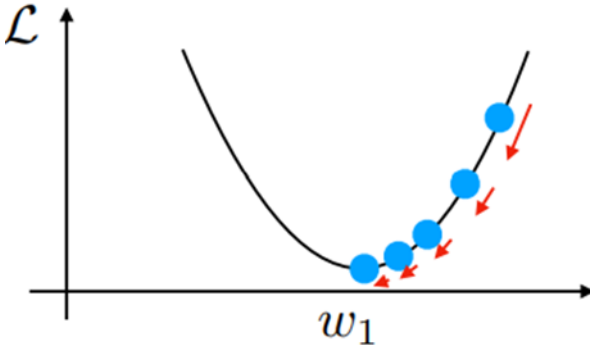
η = Learning rate

controla el tamaño del paso

- Demasiado grande → diverge
- Demasiado pequeño → lento







Gradiente como regla de actualización

Gradiente descendente estocástico

Result: $\mathbf{w}, b$
$\mathbf{w} := \mathbf{0} \in \Re^m, \mathbf{b} := 0;$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    $\hat{y}^{[i]} := \sigma(\mathbf{x}^{[i]\top}\mathbf{w} + b);$
    $\Delta_{\mathbf{w}}\mathcal{L} = (y^{[i]} - \hat{y}^{[i]})\mathbf{x}^{[i]}$;
    $\Delta_b\mathcal{L} = (y^{[i]} - \hat{y}^{[i]});$
    $\mathbf{w} := \mathbf{w} + \eta \times (-\Delta_{\mathbf{w}}\mathcal{L})$;
    $b : b + \eta \times (-\Delta_b\mathcal{L});$
  end
end

x	y
50	150
80	200
120	280

Inicialización→
w = 0
b = 0
η = 0.0001

Iteración	Época	Dato	w	b
1	1	50,150	0.750	0.015
2	1	80, 200	1.869	0.028
3	1	120,280	2.536	0.034

Época 10→

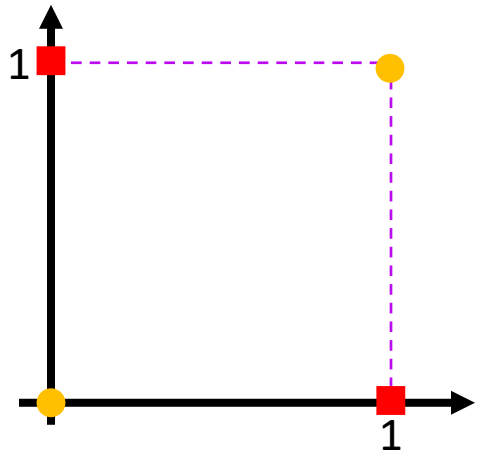
w	b
2.26	0.051

El mismo η para todos los parámetros no es óptimo

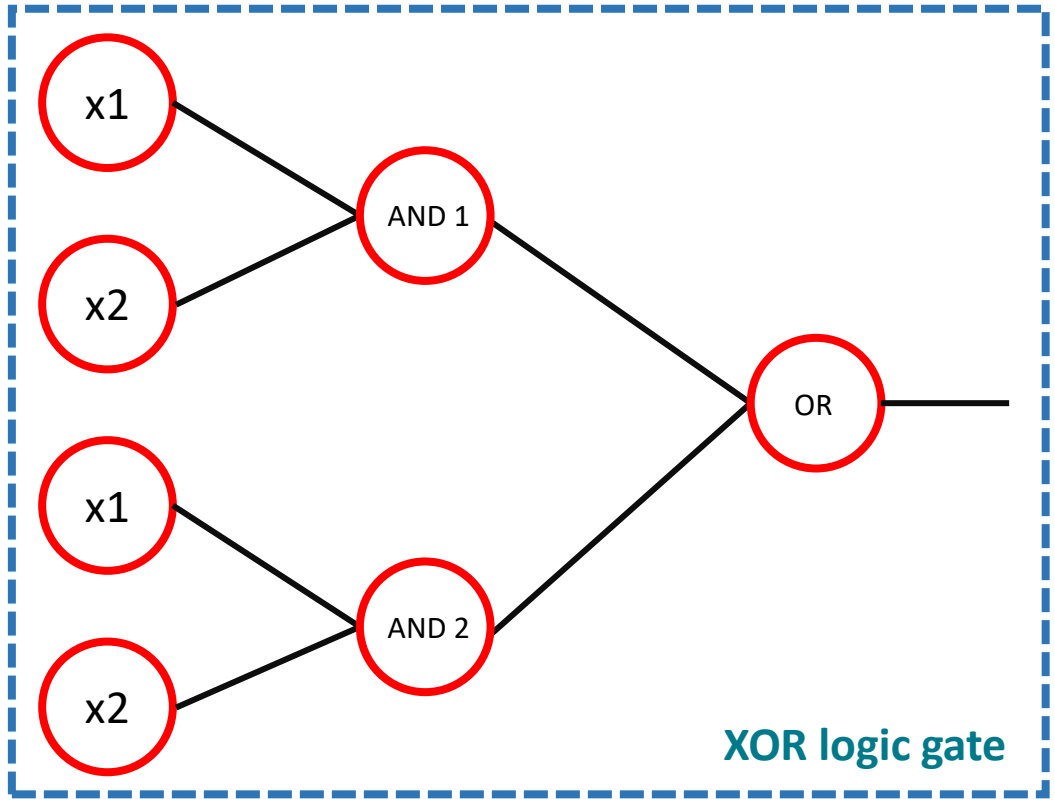
Perceptrón Multicapa MLP

XOR

Inputs		Output
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



La puerta lógica XOR se puede representar como la combinación de puertas lógicas básicas. XOR → OR[AND1(x_1, $\bar{x}_2$); AND1($\bar{x}_1$, x_2)]



AND

Inputs		Output
x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

OR

Inputs		Output
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

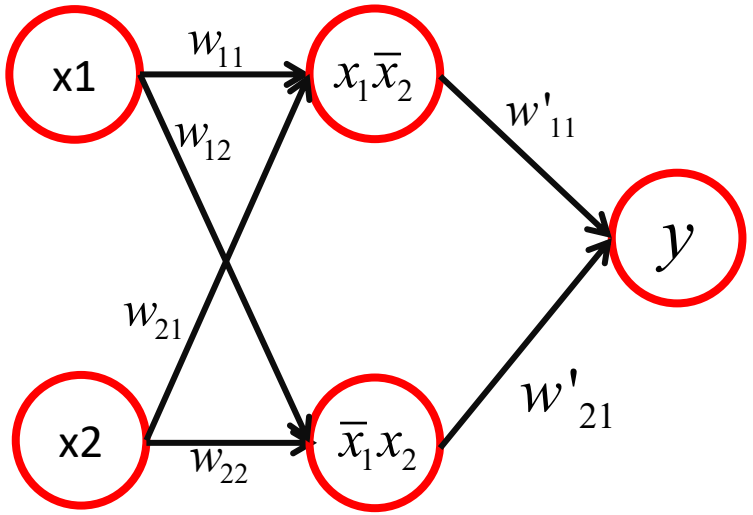
Perceptrón Multicapa MLP

OR[AND1(x_1, $\bar{x}_2$); AND1($\bar{x}_1$, x_2)]

Inputs		Layers			Output
x1	x2	AND 1	AND 2	OR	XOR
0	0	0	0	0	0
0	1	0	1	1	1
1	0	1	0	1	1
1	1	0	0	0	0

$$y = x_1\bar{x}_2 + \bar{x}_1 x_2$$

γ=1.5 $\sigma\{..\}$ → 1; if .. ≥ / 0; if ..< 1



$$f_1 = w_{ij} \times x_i \rightarrow \quad f_1 = x_1\bar{x}_2 \quad w_{11} = 1 \quad w_{21} = 1$$

$[0,0] \rightarrow 1(0)+1(0) = 0 \rightarrow \sigma[0] = 0$

$[0,1] \rightarrow 1(0)+1(1) = 1 \rightarrow \sigma[1] = 1$ ✗

$[1,0] \rightarrow 1(1)+1(0) = 1 \rightarrow \sigma[1] = 1$

$[1,1] \rightarrow 1(1)+1(1) = 2 \rightarrow \sigma[2] = 1$ ✗

La segunda y última salida no concuerda con la salida AND1. Por lo tanto, se requiere actualizar los valores de los pesos.

$$w'_{ij} = w_{ij} + \gamma(y - \hat{y})x_i$$

$$w'_{11} = 1 + 1.5(0-1)0 = 1 \qquad w'_{21} = 1 + 1.5(0-1)1 = -0.5$$

$[0,0] \rightarrow 1(0) - 0.5(0) = 0 \rightarrow \sigma[0] = 0$

$[0,1] \rightarrow 1(0) - 0.5(1) = -0.5 \rightarrow \sigma[-1.5] = 0$

$[1,0] \rightarrow 1(1) - 0.5(0) = 1 \rightarrow \sigma[1] = 1$

$[1,1] \rightarrow 1(1) - 0.5(1) = 0.5 \rightarrow \sigma[0.5] = 0$

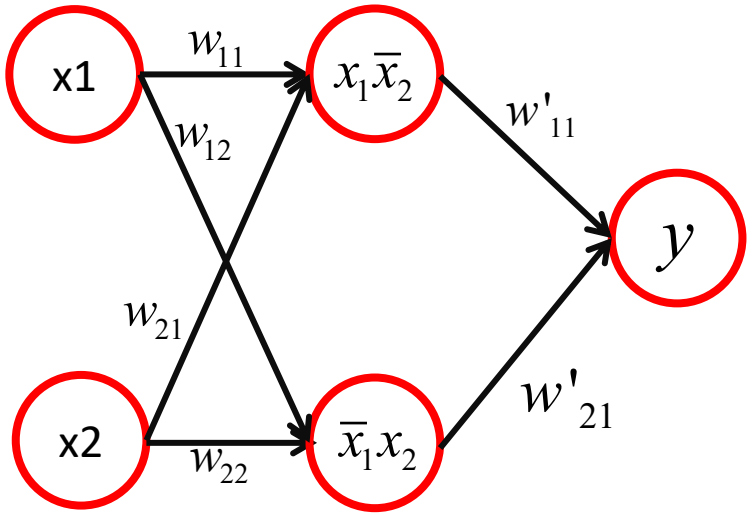
Perceptrón Multicapa MLP

OR[AND1(x_1, $\bar{x}_2$); AND1($\bar{x}_1$, x_2)]

Inputs		Layers			Output
x1	x2	AND 1	AND 2	OR	XOR
0	0	0	0	0	0
0	1	0	1	1	1
1	0	1	0	1	1
1	1	0	0	0	0

$$y = x_1\bar{x}_2 + \bar{x}_1x_2$$

γ=1.5 $\sigma\{..\}$ → 1; if .. ≥ / 0; if ..< 1



$$f_2 = w_{ij} \times x_i \rightarrow [0,0] \rightarrow 1(0)+1(0) = 0 \rightarrow \sigma[0] = 0$$
$$f_2 = \bar{x}_1x_2 \quad [0,1] \rightarrow 1(0)+1(1) = 1 \rightarrow \sigma[1] = 1$$
$$w_{12} = 1 \quad [1,0] \rightarrow 1(1)+1(0) = 1 \rightarrow \sigma[1] = 1 \text{ ✗}$$
$$w_{22} = 1 \quad [1,1] \rightarrow 1(1)+1(1) = 2 \rightarrow \sigma[2] = 1 \text{ ✗}$$

La tercera y última salida no concuerda con la salida AND2. Por lo tanto, se requiere actualizar los valores de los pesos.

$$w'_{ij} = w_{ij} + \gamma(y - \hat{y})x_i$$

$$w_{12} = 1 + 1.5(0-1)1 = -0.5 \qquad w_{22} = 1 + 1.5(0-1)1 = -0.5$$

$$[0,0] = -0.5(0) - 0.5(0) = 0 \rightarrow \sigma[0] = 0$$
$$[0,1] = -0.5(0) - 0.5(1) = -0.5 \rightarrow \sigma[-0.5] = 0 \text{ ✗}$$
$$[1,0] = -0.5(1) - 0.5(0) = -0.5 \rightarrow \sigma[-0.5] = 0$$
$$[1,1] = -0.5(1) - 0.5(1) = -1 \rightarrow \sigma[-1] = 0$$

UNIVERSIDAD EAFIT

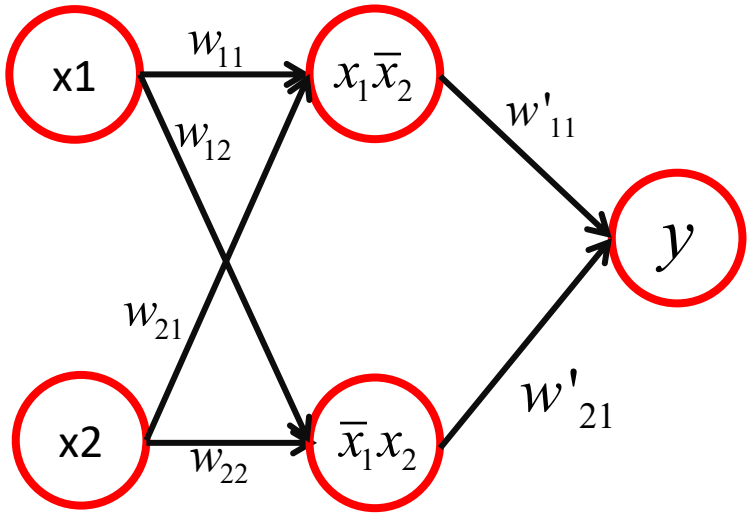
Perceptrón Multicapa MLP

OR[AND1(x_1, $\bar{x}_2$); AND1($\bar{x}_1$, x_2)]

Inputs		Layers			Output
x1	x2	AND 1	AND 2	OR	XOR
0	0	0	0	0	0
0	1	0	1	1	1
1	0	1	0	1	1
1	1	0	0	0	0

$$y = x_1\bar{x}_2 + \bar{x}_1 x_2$$

γ=1.5 $\sigma\{..\}$ → 1; if .. ≥ / 0; if ..< 1



$$w'_{ij} = w_{ij} + \gamma(y - \hat{y})x_i$$

$$w_{22} = -0.5 + 1.5(1 - 0)1 = 1$$

El perceptrón no corrige todos los pesos, corrige únicamente aquellos asociados a entradas activas que causaron el error.

$$[0,0] = -0.5(0) + 1(0) = 0 \rightarrow \sigma[0] = 0$$

$$[0,1] = -0.5(0) + 1(1) = 1 \rightarrow \sigma[1] = 1$$

$$[1,0] = -0.5(1) + 1(0) = -0.5 \rightarrow \sigma[-0.5] = 0$$

$$[1,1] = -0.5(1) + 1(1) = 0.5 \rightarrow \sigma[0.5] = 0$$

$$\begin{bmatrix} 1 & -05 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix}$$

UNIVERSIDAD EAFIT

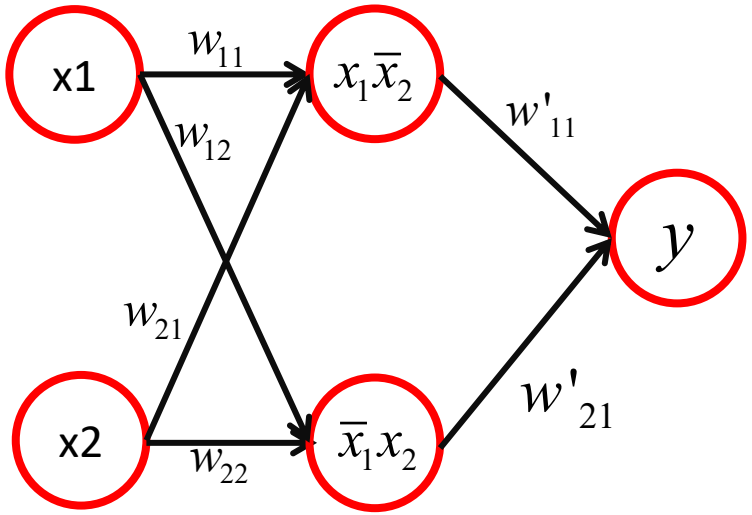
Perceptrón Multicapa MLP

OR[AND1(x_1, $\bar{x}_2$); AND1($\bar{x}_1$, x_2)]

Inputs		Layers			Output
x1	x2	AND 1	AND 2	OR	XOR
0	0	0	0	0	0
0	1	0	1	1	1
1	0	1	0	1	1
1	1	0	0	0	0

$$y = x_1\bar{x}_2 + \bar{x}_1x_2$$

γ=1.5 $\sigma\{..\}$ { 1; if .. ≥ ; 0; if ..< 1



$$f_3 = w_{ij} \times x_i \rightarrow [0,0] \rightarrow 1(0)+1(0)=0 \rightarrow \sigma[0]=0$$
$$f_3 = \bar{x}_1x_2 + x_1\bar{x}_2 \quad [0,1] \rightarrow 1(0)+1(1)=1 \rightarrow \sigma[1]=1$$
$$w'_{11}=1 \quad [1,0] \rightarrow 1(1)+1(0)=1 \rightarrow \sigma[1]=1$$
$$w'_{21}=1 \quad [0,0] \rightarrow 1(0)+1(0)=0 \rightarrow \sigma[0]=0$$

[1,1]→ $\begin{bmatrix} 1 & -05 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0.5\\0.5\end{bmatrix}$ $\sigma\{..\}\begin{bmatrix}0\\0\end{bmatrix}$

$[1\ 1]\begin{bmatrix}0\\0\end{bmatrix} = 0 + 0 = 0$

[0,1]→ $\begin{bmatrix} 1 & -05 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}-0.5\\1\end{bmatrix}$ $\sigma\{..\}\begin{bmatrix}0\\1\end{bmatrix}$

$[1\ 1]\begin{bmatrix}0\\1\end{bmatrix} = 0 + 1 = 1$

UNIVERSIDAD EAFIT

Regla de aprendizaje del perceptrón

Result: $\mathbf{w}, b$
$\mathbf{w} := \mathbf{0} \in \Re^m, \mathbf{b} := 0;$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    $\hat{y}^{[i]} := \sigma(\mathbf{x}^{[i]\top}\mathbf{w} + b);$
    $err := (y^{[i]} - \hat{y}^{[i]});$
    $\mathbf{w} := \mathbf{w} + err \times \mathbf{x}^{[i]},$
    $b := b + err;$
  end
end

Gradiente descendente estocástico

Result: $\mathbf{w}, b$
$\mathbf{w} := \mathbf{0} \in \Re^m, \mathbf{b} := 0;$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    $\hat{y}^{[i]} := \sigma(\mathbf{x}^{[i]\top}\mathbf{w} + b);$
    $\Delta_{\mathbf{w}}\mathcal{L} = (y^{[i]} - \hat{y}^{[i]})\mathbf{x}^{[i]}$;
    $\Delta_b\mathcal{L} = (y^{[i]} - \hat{y}^{[i]});$
    $\mathbf{w} := \mathbf{w} + \eta \times (-\Delta_{\mathbf{w}}\mathcal{L})$; gradiente negativo
    $b : b + \eta \times (-\Delta_b\mathcal{L});$
  end
end

tasa de aprendizaje

Vectorizado

Result: $\mathbf{w}, b$
$\mathbf{w} := \mathbf{0} \in \Re^m, \mathbf{b} := 0;$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    $\hat{y}^{[i]} := \sigma(\mathbf{x}^{[i]\top}\mathbf{w} + b);$
    $\Delta_{\mathbf{w}}\mathcal{L} = (y^{[i]} - \hat{y}^{[i]})\mathbf{x}^{[i]}$;
    $\Delta_b\mathcal{L} = (y^{[i]} - \hat{y}^{[i]});$
    $\mathbf{w} := \mathbf{w} + \eta \times (-\Delta_{\mathbf{w}}\mathcal{L})$;
    $b : b + \eta \times (-\Delta_b\mathcal{L});$
  end
end

Ciclo *for*

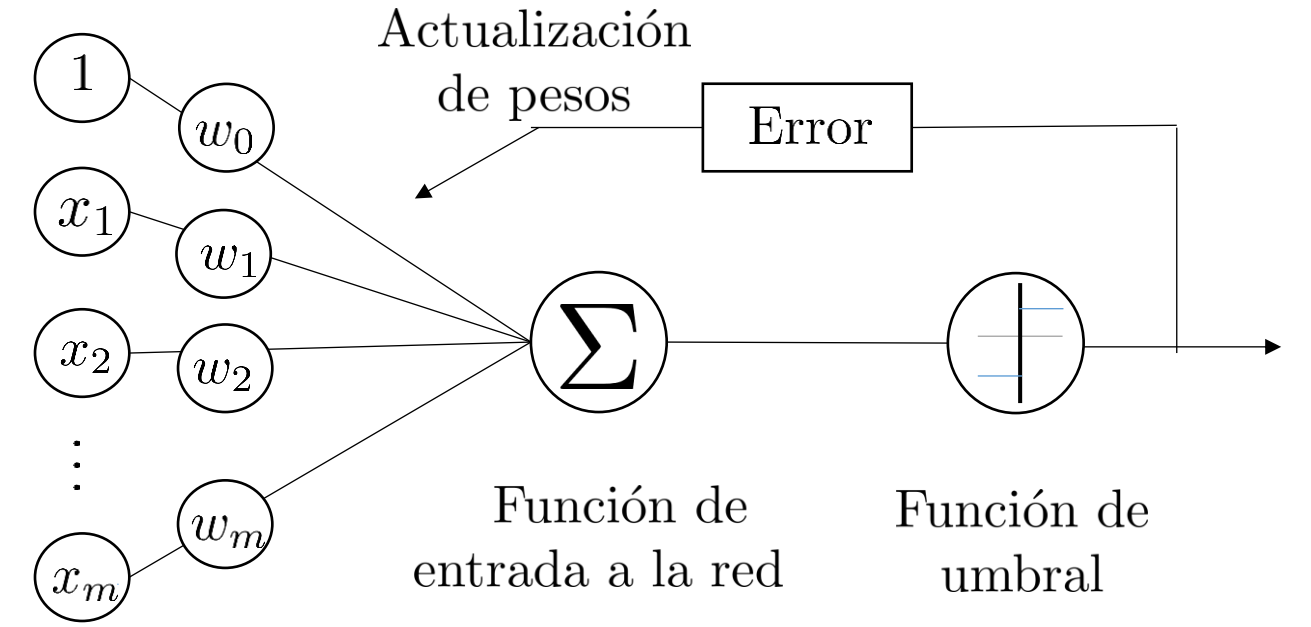
Result: $\mathbf{w}, b$
$\mathbf{w} := \mathbf{0} \in \Re^m, \mathbf{b} := 0;$
for $t = 1, \ldots, T$ **do**
  $\Delta\mathbf{w} := 0, \Delta b := 0;$
  for $i = 1, \ldots, n$ **do**
    $\hat{y}^{[i]} := \sigma(\mathbf{x}^{[i]\top}\mathbf{w} + b);$
  end
  for $j = 1, \ldots, m$ **do**
    $\frac{\partial \mathcal{L}}{\partial w_j} = (y^{[i]} - \hat{y}^{[i]})x_j^{[i]};$
    $w_j := w_j + \eta \times \left(-\frac{\partial \mathcal{L}}{\partial w_j}\right);$
  end
  $\frac{\partial \mathcal{L}}{\partial b} = (y^{[i]} - \hat{y}^{[i]});$
  $b := b + \eta \times \left(-\frac{\partial \mathcal{L}}{\partial b}\right);$
end

Modo *On-line*

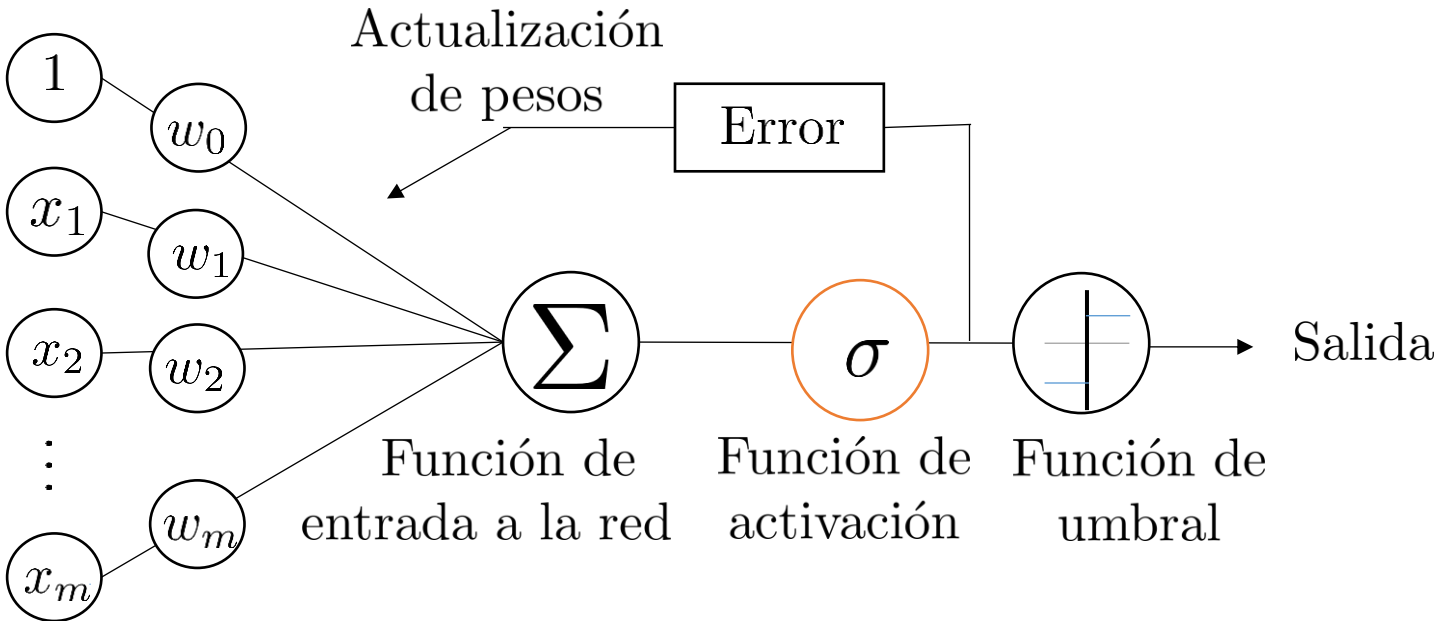
Result: $\mathbf{w}, b$
$\mathbf{w} := \mathbf{0} \in \Re^m, \mathbf{b} := 0;$
for $t = 1, \ldots, T$ **do**
  $\Delta\mathbf{w} := 0, \Delta b := 0;$
  for $i = 1, \ldots, n$ **do**
    $\hat{y}^{[i]} := \sigma(\mathbf{x}^{[i]\top}\mathbf{w} + b);$
  end
  for $j = 1, \ldots, m$ **do**
    $\frac{\partial \mathcal{L}}{\partial w_j} = (y^{[i]} - \hat{y}^{[i]})x_j^{[i]};$
    $w_j := w_j + \eta \times \left(-\frac{\partial \mathcal{L}}{\partial w_j}\right);$
  end
  $\frac{\partial \mathcal{L}}{\partial b} = (y^{[i]} - \hat{y}^{[i]});$
  $b := b + \eta \times \left(-\frac{\partial \mathcal{L}}{\partial b}\right);$
end

Coincidencialmente, parece casi el mismo de la regla del perceptrón, exceptuando que la predicción es un número real y se tiene una tasa de aprendizaje. Esta regla de aprendizaje se llama **gradiente descendente estocástico**

Redes Neuronales



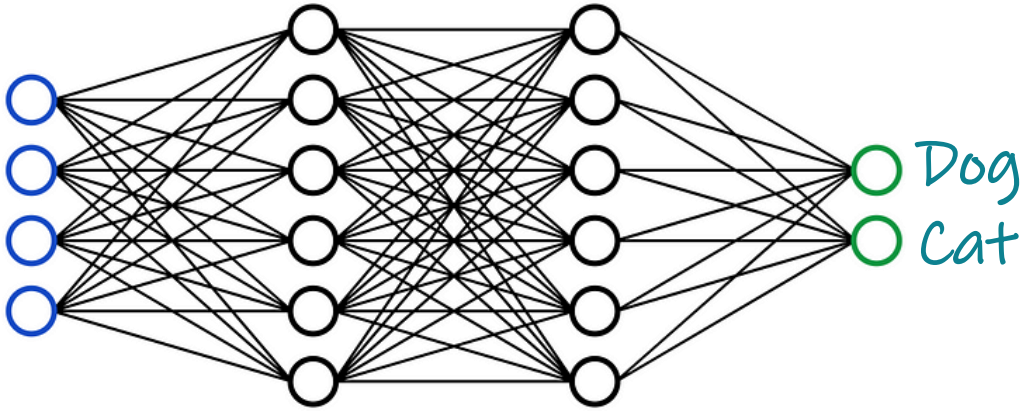
Perceptrón

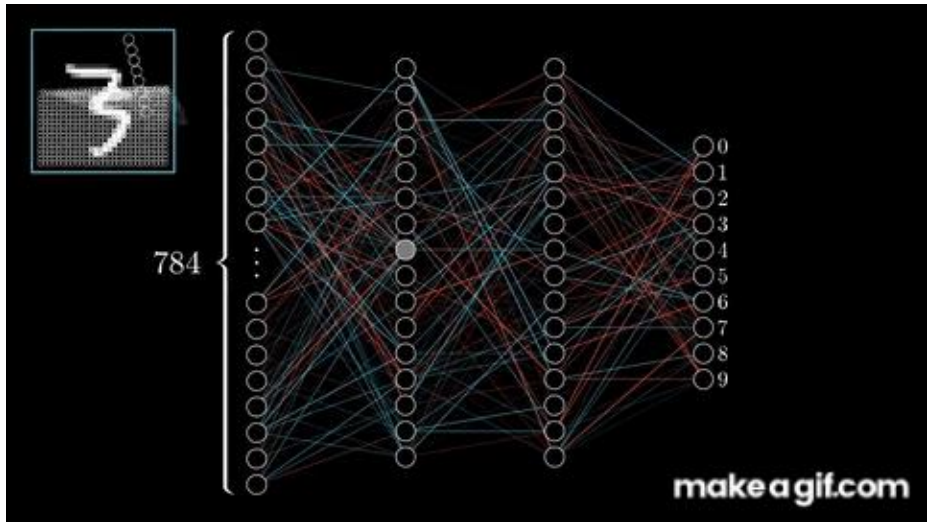


ADAptive LInear NEuron

UNIVERSIDAD EAFIT

Redes Neuronales



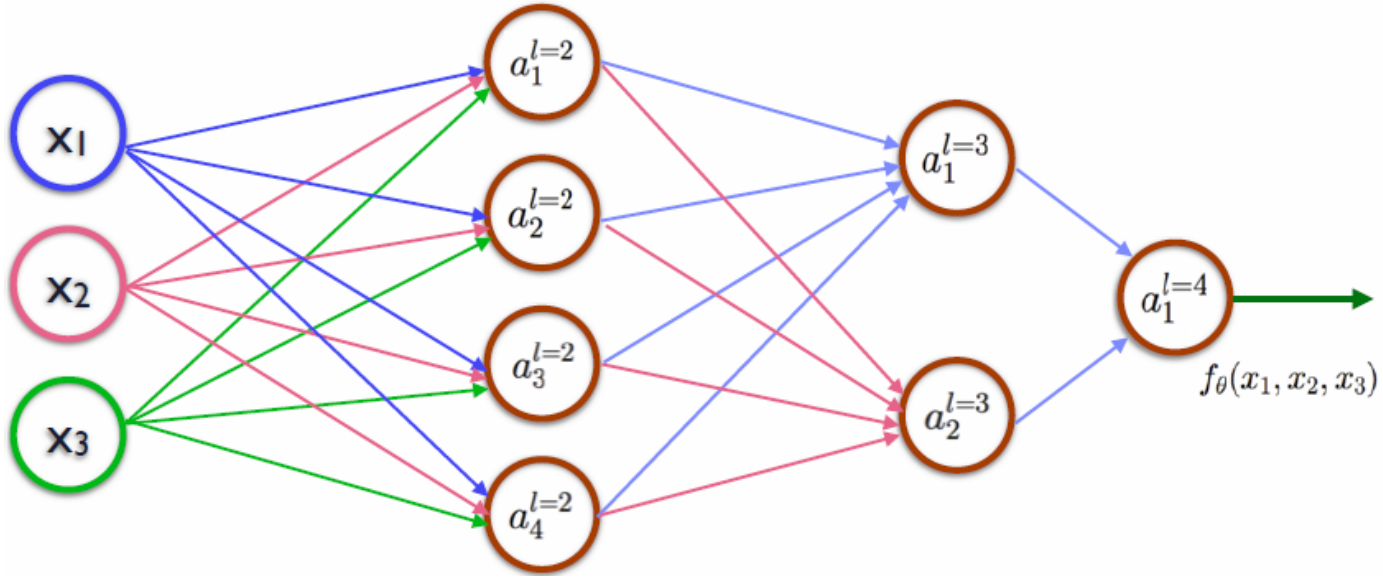


Input data
(Features)

$$\vec{x} = \{x_1, x_2, x_3, ...x_n\}$$

Output data
(predictions - labels)

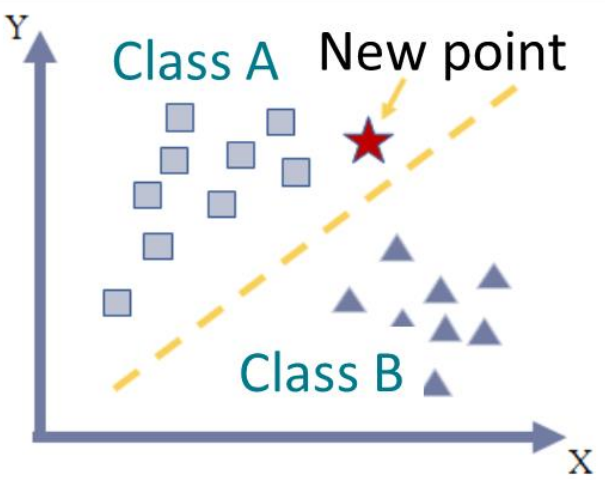
$$\hat{\vec{y}} = \{\hat{y}_1, \hat{y}_2, \hat{y}_3, ...\hat{y}_n\}$$

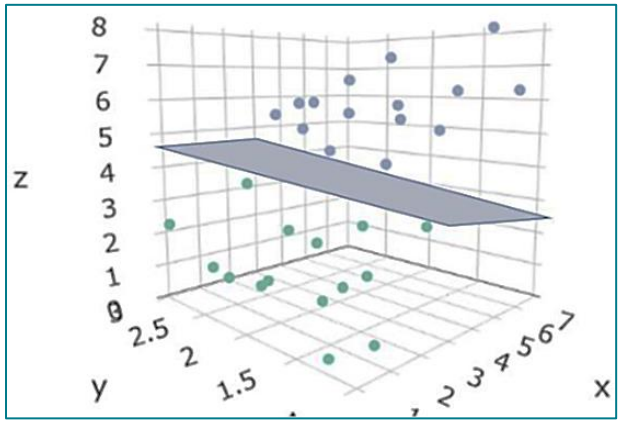


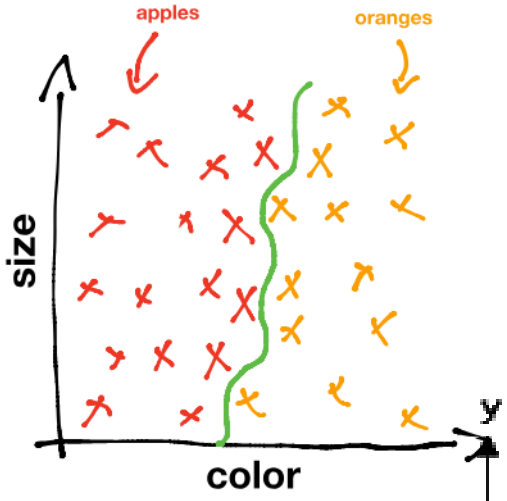
UNIVERSIDAD EAFIT

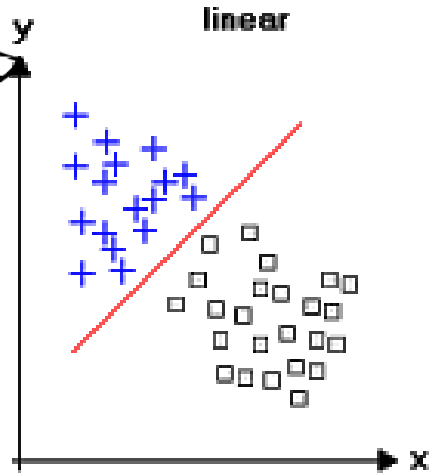
Regresión Logística

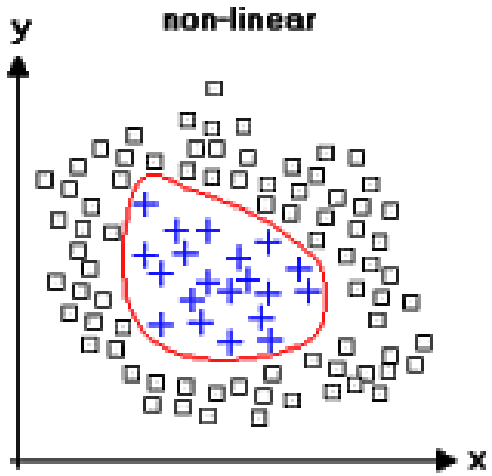
Método de clasificación binaria

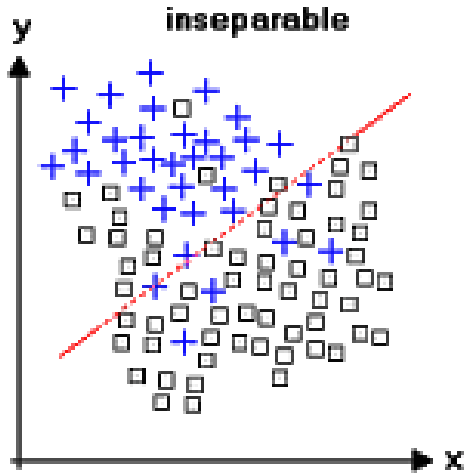




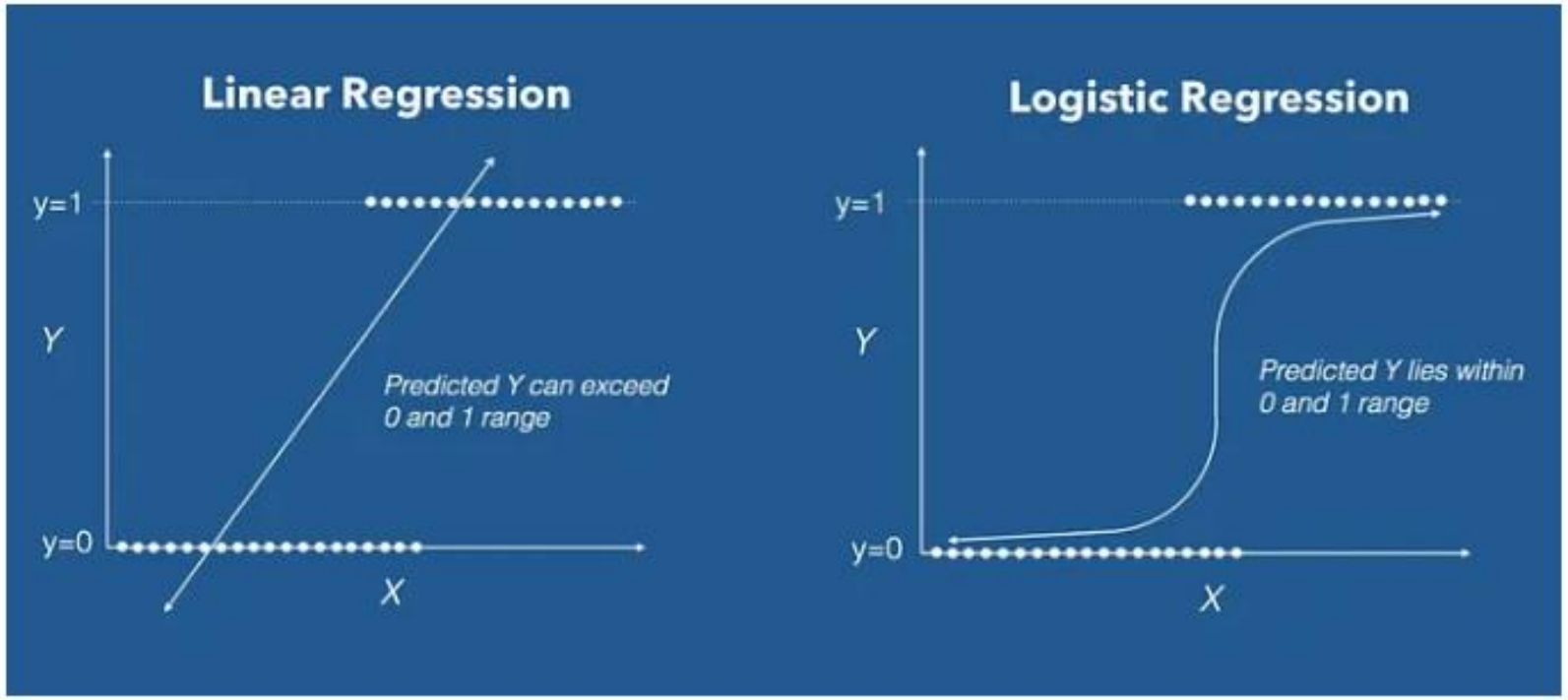








Regresión Logística

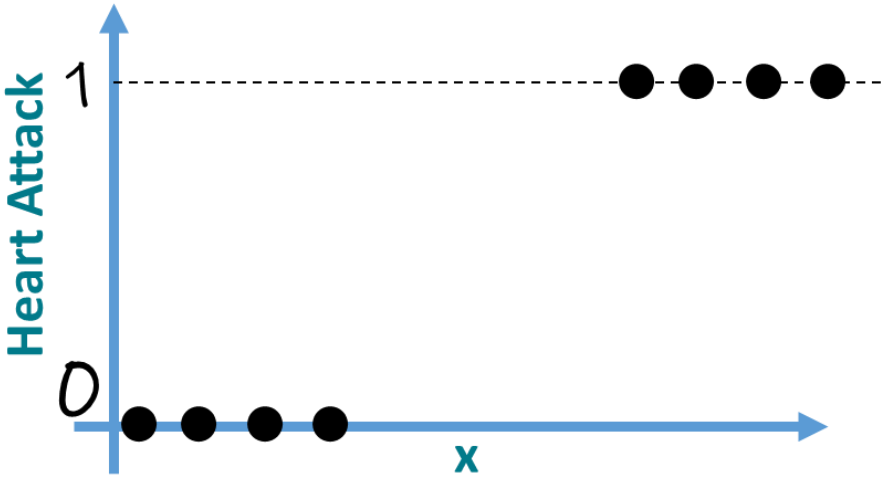


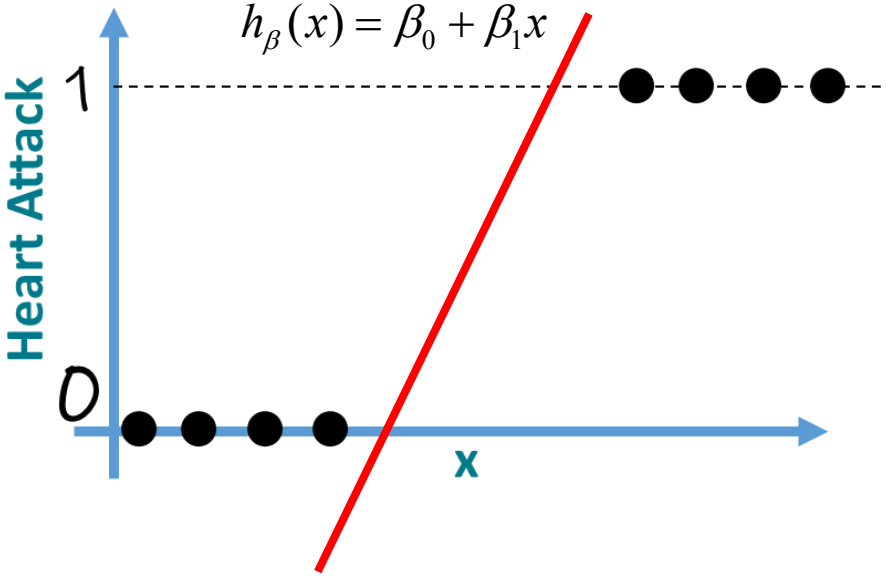
Linear Regression VS Logistic Regression Graphl Image: Data Camp

La regresión logística predice una probabilidad continua entre 0 y 1, que luego se convierte en una decisión binaria mediante un umbral.

La regresión logística se deriva **directamente** del principio de **Máxima Verosimilitud (MLE)**. **Maximum Likelihood**

Regresión Logística





Original data

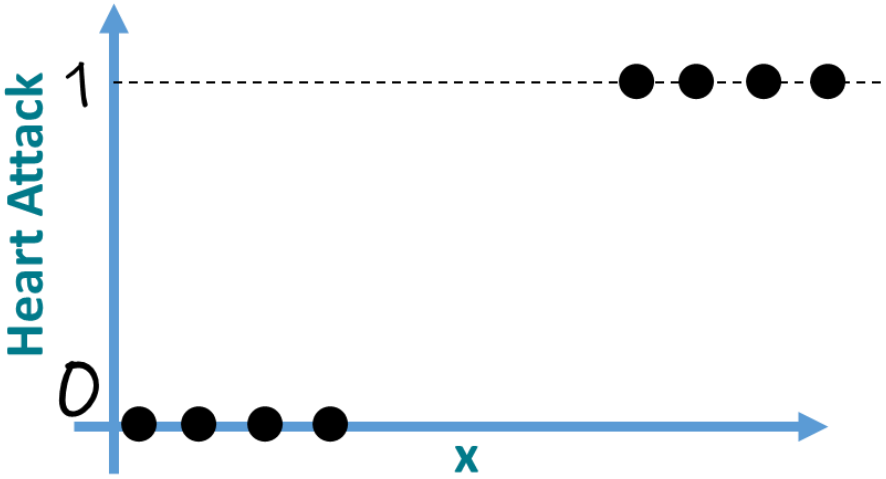
New data

0 1

Negative class Positive class

X = Smokes a lot and drinks alcohol

Regresión Logística



Original data

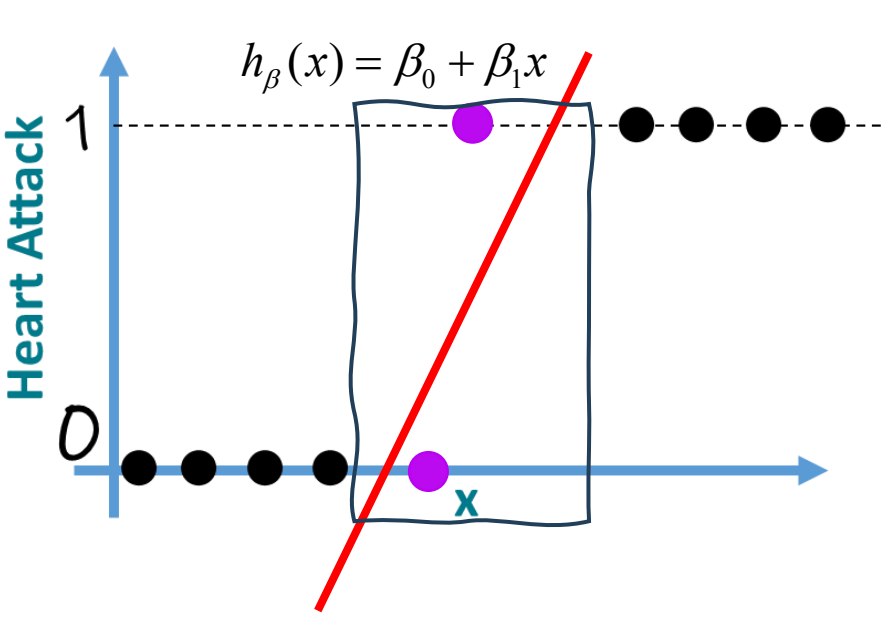
New data

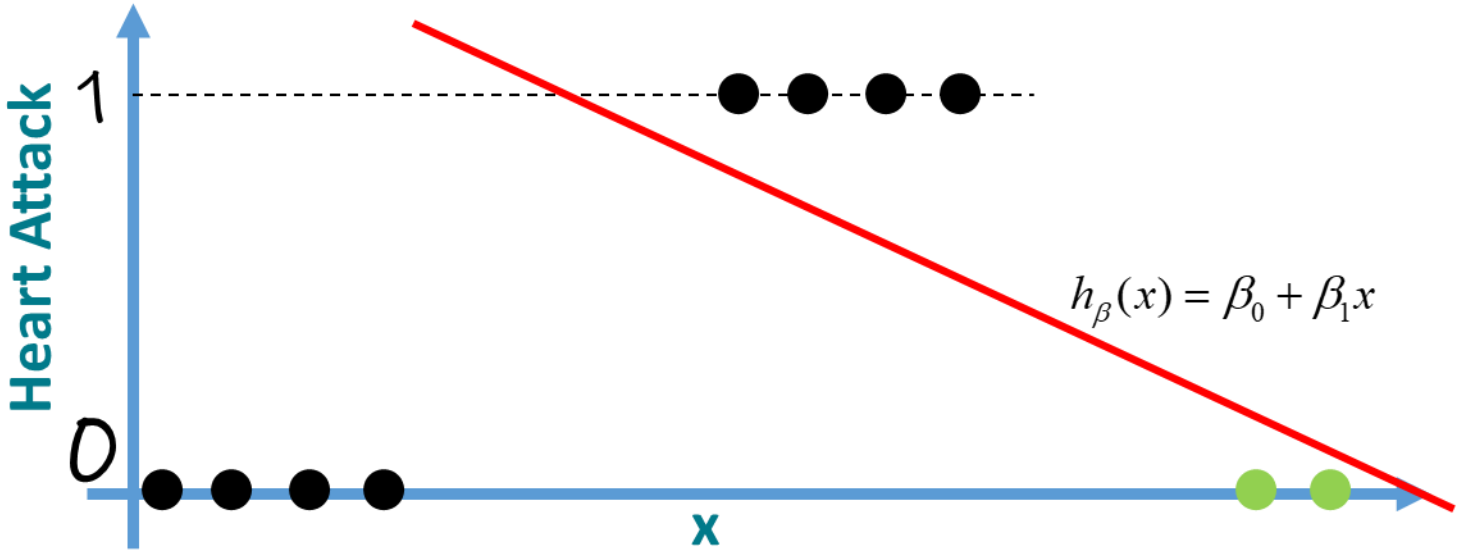
Outliers

0 1

Negative class Positive class

X = Smokes a lot and drinks alcohol





Regresión Logística

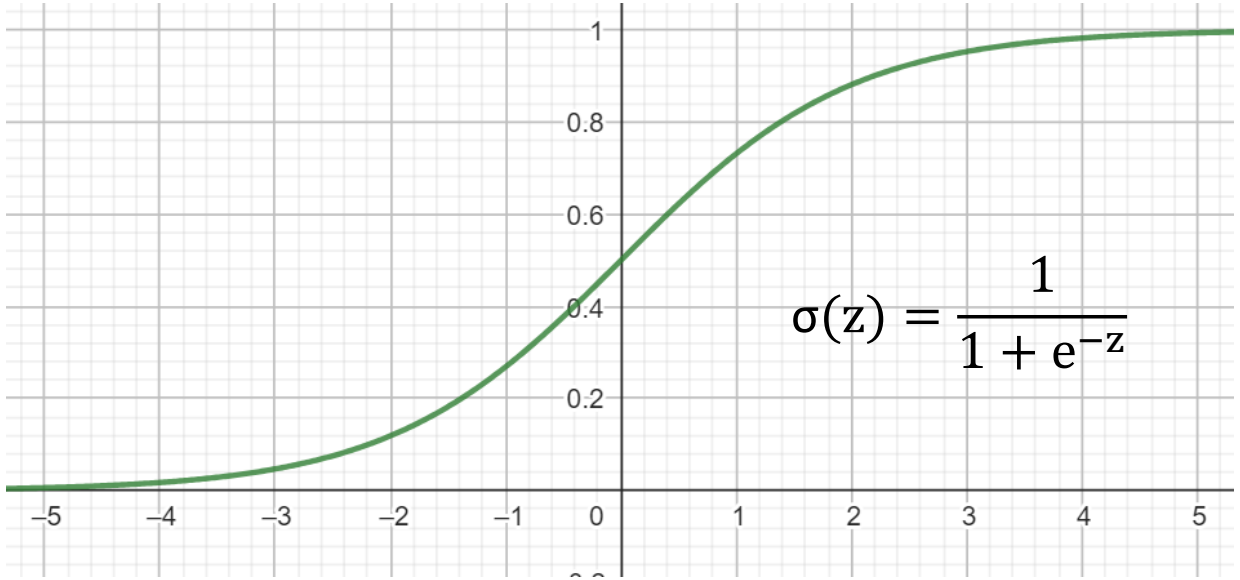
- La regresión logística exige una función que convierta valores en [0 , 1]

$0 < \sigma(z) < 1;\ \forall\ z \in R$

$\sigma(z) \in (0, 1)$

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}} \begin{cases} sigmoid \geq 0.5 \rightarrow 1 \\ sigmoid < 0.5 \rightarrow 0 \end{cases}$$

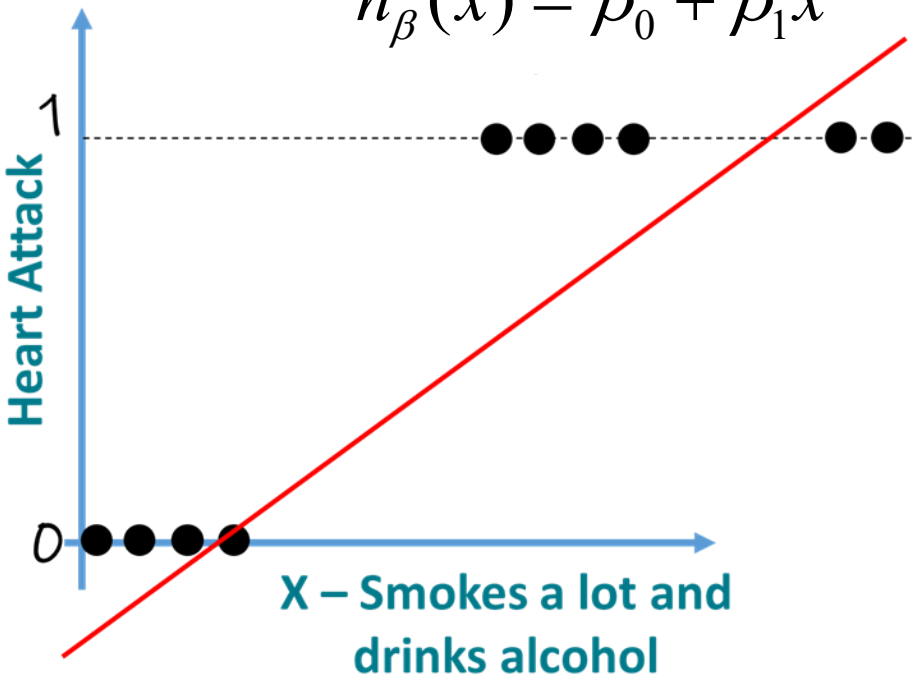
La función sigmoide toma un valor de entrada y lo mapea a un valor comprendido entre 0 y 1



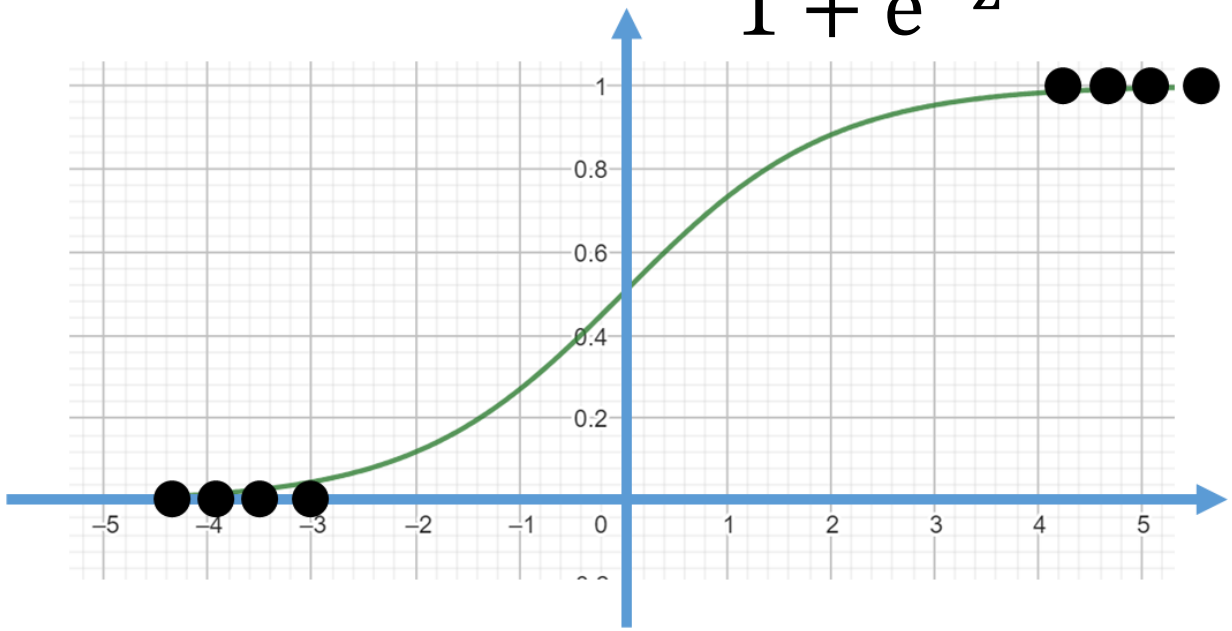
Es importante destacar que el objetivo de la regresión logística no es, como en la regresión lineal, predecir el valor de la variable Y a partir de una o varias variables predictoras, sino estimar la probabilidad de que Y ocurra dado el valor de dichas variables

Regresión Logística

$$h_\beta(x) = \beta_0 + \beta_1 x$$



$$\text{sigmoid(z)} = \frac{1}{1 + e^{-z}}$$



$$\sigma(z) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$p_i \rightarrow$ Probabilidad de que el evento ocurra.

P (Y = 1| x) = p

$1 - p_i \rightarrow$ Probabilidad de que el evento no ocurra.

$\frac{p_i}{1 - p_i} \rightarrow$ Razón de Probabilidad.

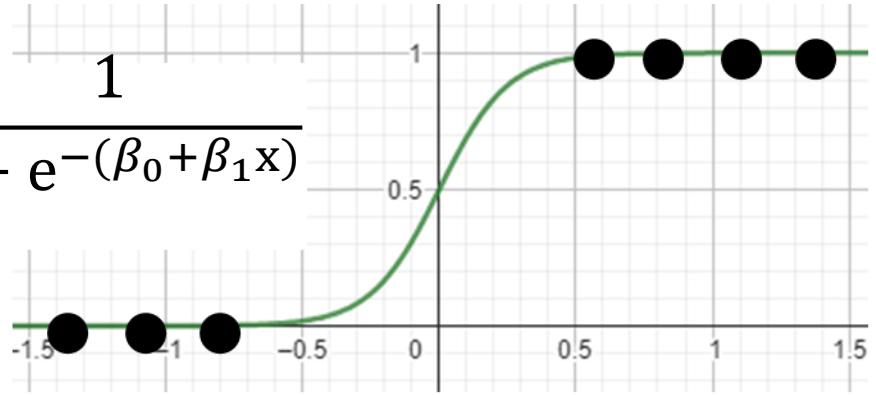
Regresión Logística

P (Y = 1| x) = p

P (Y = 0| x) = 1 – p

La probabilidad de que la variable aleatoria Y tome el valor 1, dado el vector de características x, es igual a p.

$$\sigma(z) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

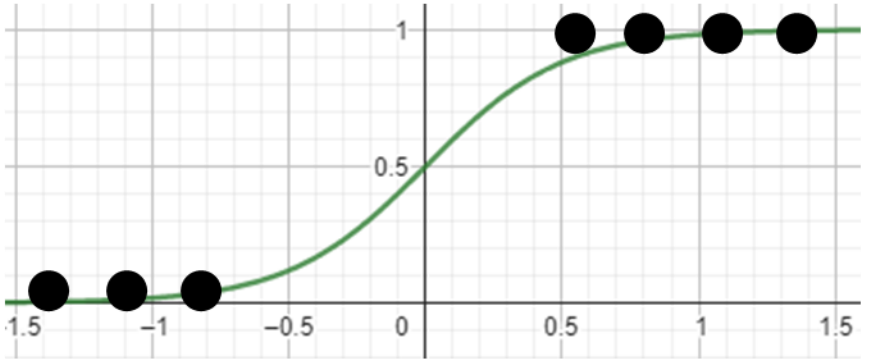


$Y \in \{0, 1\}$

$x = (x_1, x_2, x_3, \ldots . x_n)$

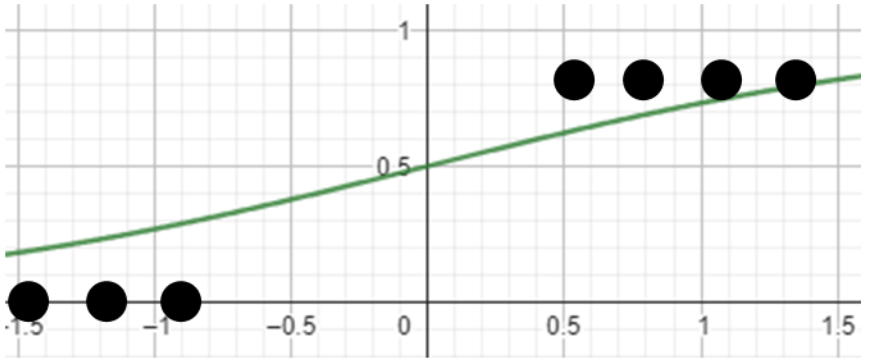
La regresión logística no predice directamente si el evento ocurre o no, sino qué tan probable es que ocurra, dadas las características de entrada.

$P(Y = 1| x) = \sigma(\beta_0 + \beta^T x)$



$$Ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon$$

$$Ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta x \Rightarrow \frac{p}{1-p} = e^{\beta_0 + \beta x}$$



Cuál es la manera optimizada de encontrar $[\beta_0, \beta]$?

Regresión Logística

$$P(Y = 1|X = x; \beta) = h_b(x)$$

$$P(Y = 0|X = x; \beta) = 1 - h_b(x)$$

$$P(Y = y|X = x; \beta) = h_b(x)^y(1 - h_b(x))^{1-y}$$

$$P(Y = y|X = x; \beta) = \sigma(\beta_0 + \beta^T x)^y[1 - \sigma(\beta_0 + \beta^T x)]^{1-y}$$

- Probabilidad con distribución Bernoulli ya que la variable aleatoria condicionada es binaria.
- Se asume que los datos son mutuamente independientes.
- Consideremos un conjunto de datos $D\{(x_i, y_i)\}_{i=1}^N$

Vector de características

Etiquetas

Bernoulli

$$P(Y = 1|X = x) = p(x)$$

$$P(Y = 0|X = x) = 1 - p(x)$$

$$P(Y = y|X = x; \beta) = p(x)^y[1 - p(x)]^{1-y}$$

$$D = \{(x_1, y_1); (x_2, y_2); \dots\dots; (x_N, y_N)\}$$

$$p_i = \sigma(z_i)$$

Likelihood

- Probabilidad de observar todas las etiquetas del dataset, dadas las entradas y los parámetros:

$$L(\beta) = P(D|\beta)$$

- Asumiendo que las observaciones son independientes $$L(\beta) = P(D|\beta) = \prod_{i=1}^{N} P(\mathrm{Y_i} = \mathrm{y_i}|\mathrm{X_i} = \mathrm{x_i}; \beta)$$

- La probabilidad conjunta del dataset es el producto de las probabilidades individuales

- La verosimilitud de βes el producto, para cada dato i, de la probabilidad asignada a la clase correcta. $$L(\beta) = P(D|\beta) = \prod_{i=1}^{N} \mathrm{p(x)^y}[1 - \mathrm{p(x)}]^{1-\mathrm{y}}$$

$$p_i = \sigma(z_i)$$

$$P(\mathrm{Y_i} = \mathrm{y_i}|\mathrm{X_i} = \mathrm{x_i}; \beta) = \sigma(z_i)^{\mathrm{y}_i}[1 - \sigma(z_i)]^{1-\mathrm{y}_i}$$

$$L(\beta) = P(D|\beta) = \prod_{i=1}^{N} \sigma(z_i)^{\mathrm{y}_i}[1 - \sigma(z_i)]^{1-\mathrm{y}_i}$$

Regresión Logística

$$\ln[L(\beta)] = l(\beta)$$

$$l(\beta) = \prod_{i=1}^{N} \sigma(z_i)^{y_i}[1 - \sigma(z_i)]^{1-y_i}$$

$$l(\beta) = \sum_{i=1}^{N} ln(\sigma(z_i)^{y_i}[1 - \sigma(z_i)]^{1-y_i})$$

$$l(\beta) = \sum_{i=1}^{N} [\ln(\sigma(z_i)^{y_i}) + ln([1 - \sigma(z_i)]^{1-y_i})]$$

$$l(\beta) = \sum_{i=1}^{N} [y_i ln(\sigma(z_i)) + (1 - y_i) ln(1 - \sigma(z_i))]$$

El modelo se entrena minimizando la entropía cruzada binaria, equivalente al negativo de la log-verosimilitud Bernoulli.

$$ln\left(\prod_i a_i\right) = \sum_i ln\,(a_i)$$

$$ln(a^b) = b ln(a)$$

$$ln(ab) = ln(a) + ln(b)$$

Entropía cruzada (cross-entropy binaria)

$$J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}[y_i ln(\sigma(z_i)) + (1 - y_i) ln(1 - \sigma(z_i))]$$

Regresión Logística

$$J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}[y_i ln(\sigma(z_i)) + (1 - y_i) ln(1 - \sigma(z_i))]$$

$$\nabla J(\beta) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)x_i \qquad \frac{\partial J}{\partial \beta_0} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)$$

$\hat{y}_i - y_i$

- > 0 Si el modelo sobreestima la probabilidad
- < 0 Si el modelo subestima la probabilidad
- = 0 Si el modelo acierta exactamente

$$\beta \leftarrow \beta - \eta(\hat{y} - y)x$$

Caso 1 → $y_i=1$

$\hat{y}_i - 1$ $\hat{y}_i < 1$ $\hat{y}_i - 1 < 0$

Caso 2 → $y_i=0$

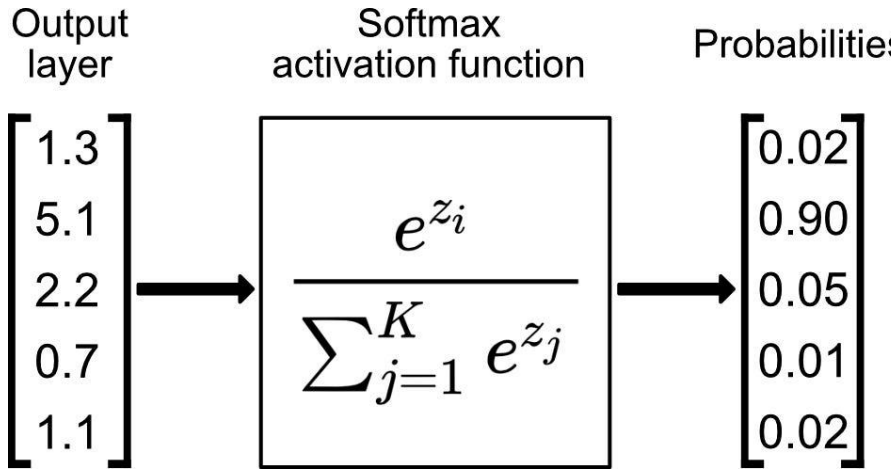
$\hat{y}_i$ $\hat{y}_i > 0$

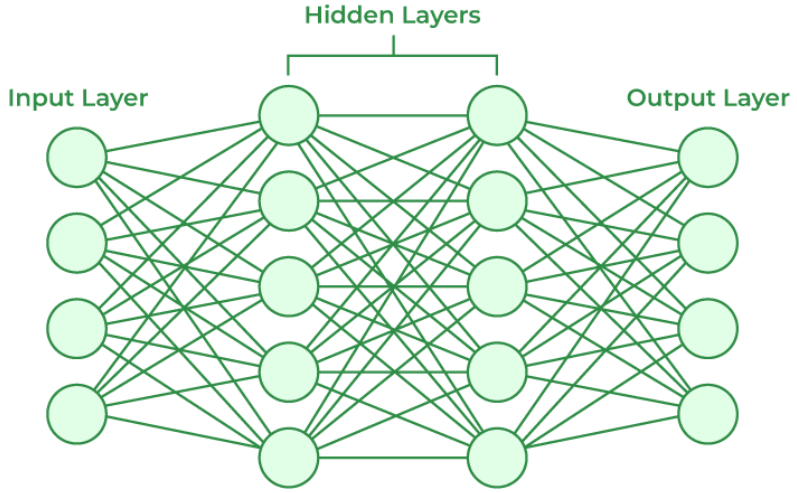
Regresión Logística Multiclase

Regresión *softmax*

$$softmax(e^{z_i}) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Queremos una sola clase ganadora
Queremos probabilidades interpretables
Funciona perfectamente con Cross-Entropy Loss



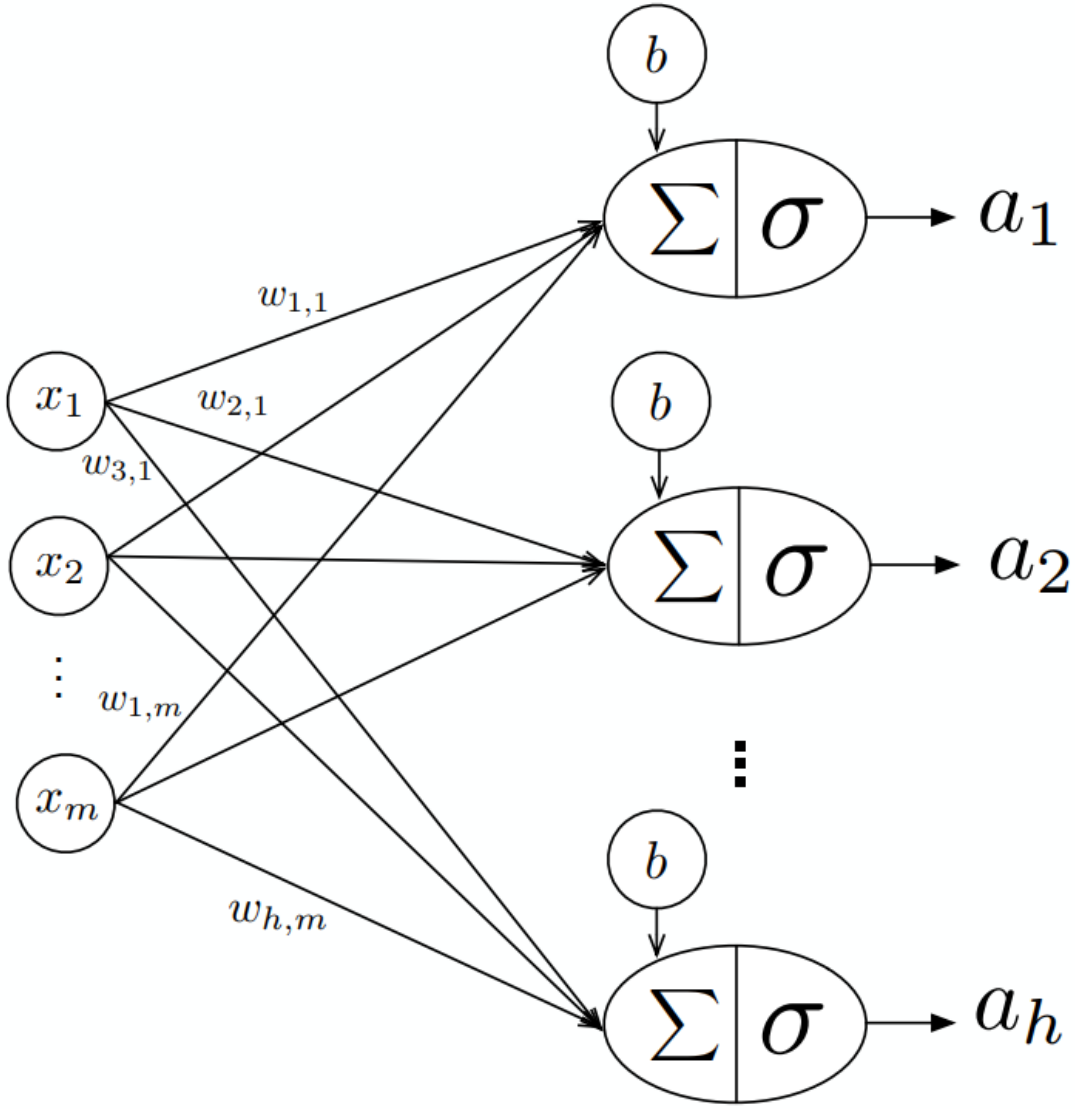


Regresión Logística Multiclase

Regresión *softmax*

En regresión *softmax*, la capa de salida de la red tiene varios nodos, uno por clase.

Las activaciones se pueden ver como proba-bilidad de perte-nencia a cada clase (no mutuamente ex-cluyente)



Regresión Logística Multi clase

Matemáticamente se tiene que:

$\mathbf{a} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \Re^{K\times 1}$, con

$\mathbf{W} \in \Re^{m\times K}$

$\mathbf{b} \in \Re^{K\times 1}$

$\mathbf{x} \in \Re^{m\times 1}$

Cada a_k será:

$a_k = \sigma(\mathbf{w}_k \cdot \mathbf{x} + b_k)$, con

$\mathbf{w}_k \in \Re^{1\times m}$ k-th fila de $\mathbf{W}$

En caso de batches $\mathbf{X} \in \Re^{N_b\times m}$

$\mathbf{A} = \sigma(\mathbf{X}\mathbf{W}^\top + \mathbf{b}) \in \Re^{N_b\times K}$

$b \in \Re^{K,}$ (Tensor 1D)

Sin embargo, para que se cumpla que a_k es una probabilidad de pertenencia a cada clase:

$\sum_{k=1}^{K} a_k = 1$

Para esto, se utiliza *softmax*

$$p(y = t|z_t^{[i]}) = \sigma_{\text{softmax}}(z_t^{[i]}) = \frac{e^{z_t^{[i]}}}{\sum_{j=1}^{h} e^{z_t^{[i]}}}$$

h : Número de clases

$t = 1, \ldots, h$

Softmax es solo una función exponencial que normaliza las activaciones para que la suma de 1

UNIVERSIDAD EAFIT

Regresión Logística Multiclase

One-hot encoding

class labels
0
1
3
2

class_0	class_1	class_2	class_3
1	0	0	0
0	1	0	0
0	0	0	1
0	0	1	0

Regresión Logística Multiclase

Función de costo

Cross-entropía multicategórica para h clases:

$$\mathcal{L} = -\sum_{i=1}^{n}\sum_{j=1}^{h} y_j^{[i]} \log(a_j^{[i]})$$

Asume etiquetas en *one-hot encoding*

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

$n = 4$

$h = 3$

$$\mathcal{L}^{[1]} = [(-1)\cdot\log(0.3792)] + [(-0)\cdot\log(0.3104)] + [(-0)\cdot\log(0.3104)] = 0.969692...$$

$$\mathcal{L}^{[2]} = [(-0)\cdot\log(0.3072)] + [(-1)\cdot\log(0.4147)] + [(-0)\cdot\log(0.2780)] = 0.880200...$$

$$\mathcal{L}^{[3]} = [(-0)\cdot\log(0.4263)] + [(-0)\cdot\log(0.2248)] + [(-1)\cdot\log(0.3490)] = 1.05268...$$

$$\mathcal{L}^{[4]} = [(-0)\cdot\log(0.2668)] + [(-0)\cdot\log(0.2978)] + [(-1)\cdot\log(0.4354)] = 0.831490...$$

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{h} -y_j^{[i]} \log\left(a_j^{[i]}\right) \approx 0.9335$$

Derivadas de regresión *softmax* por gradiente descendente

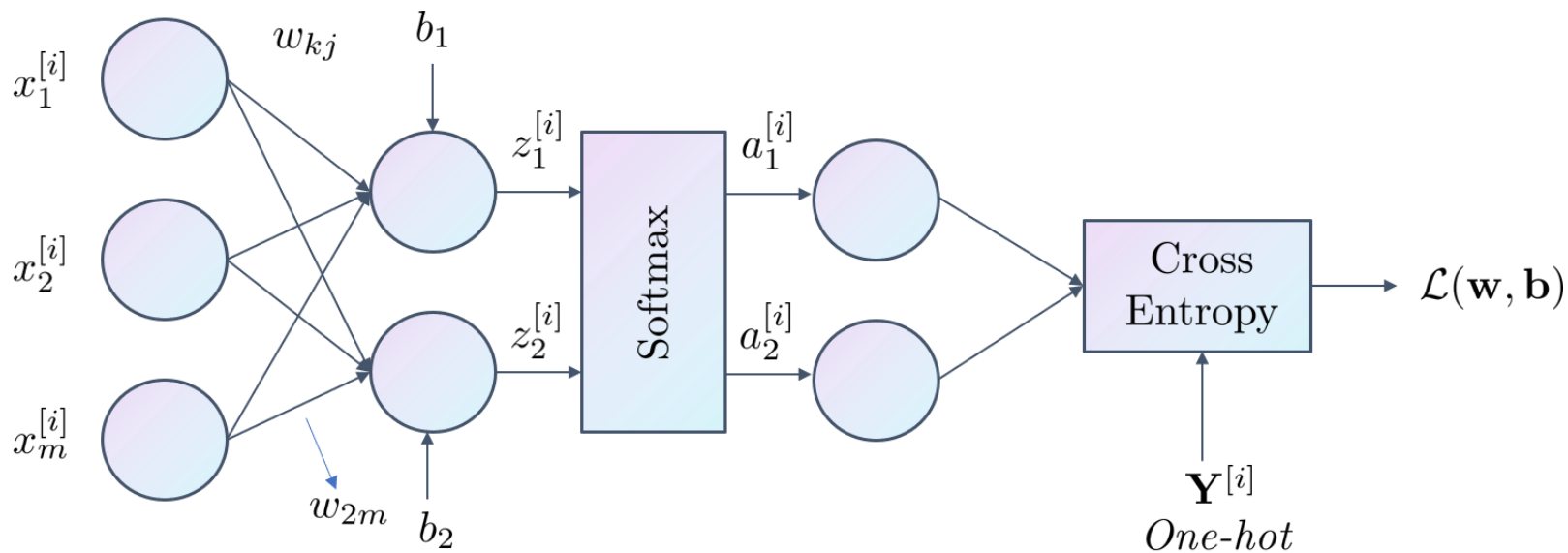
Para la regla de aprendizaje, necesitamos calcular:

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial w_{k,j}}$$

$$\frac{\partial \mathcal{L}}{\partial b_k} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial b_k}$$

$$\mathbf{x}^{[i]} \in \Re^{m \times 1}$$

$$\mathbf{X} \in \Re^{b \times m}$$



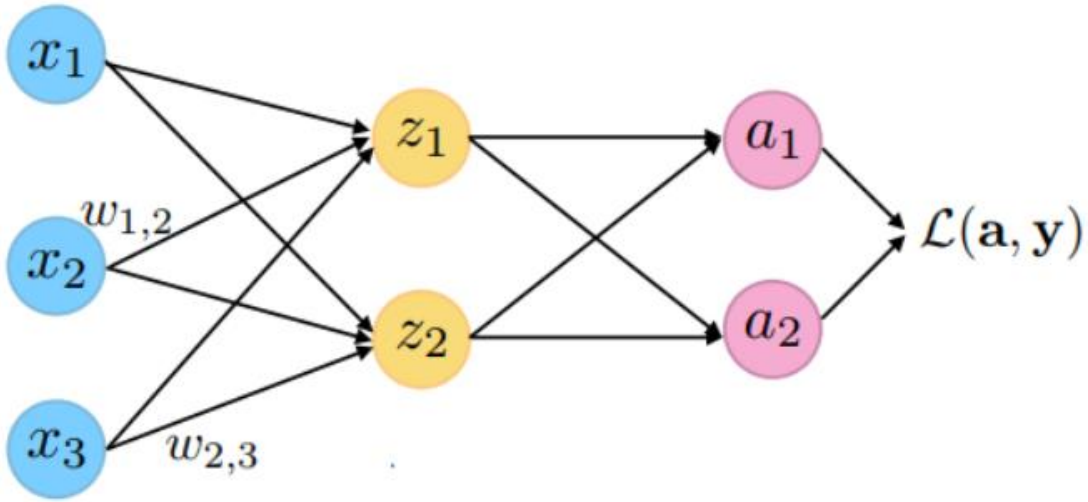
$$\mathbf{W} \in \Re^{K \times m} \qquad \mathbf{z} \in \Re^{K \times 1} \qquad \mathbf{a} \in \Re^{K \times 1}$$

$$\mathbf{Z} \in \Re^{b \times K} \qquad \mathbf{A} \in \Re^{b \times K}$$

$k = 1, \ldots, K$: llega

$j = 1, \ldots, m$: sale

Cálculo de las derivadas



$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$

$$-\frac{y_1}{a_1} \quad a_1(1-a_1) \quad x_2 \qquad -\frac{y_2}{a_2} \quad -a_2 a_1 \quad x_2$$

$$\frac{\partial L}{\partial a_1} = \frac{\partial}{\partial a_1}\left[\sum_{j=1}^{h} -y_j \log(a_j)\right]$$

$$= \frac{\partial}{\partial a_1}\left[-y_1 \log(a_1)\right]$$

$$= -\frac{y_1}{a_1}$$

$$= \frac{\partial}{\partial w_{1,2}}\left[w_{1,2} \cdot x_2 + b\right]$$

$$= x_2$$

$$\frac{\partial a_1}{\partial z_1} = \frac{\partial}{\partial z_1}\left[\frac{e^{z_1}}{\sum_{j=1}^{h} e^{z_j}}\right]$$

$$= \frac{\left[\sum_{j=1}^{h} e^{z_j}\right]\frac{\partial}{\partial z_1}e^{z_1} - e^{z_1}\frac{\partial}{\partial z_1}\left[\sum_{j=1}^{h} e^{z_j}\right]}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{\left[\sum_{j=1}^{h} e^{z_j}\right]e^{z_1} - e^{z_1}e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{e^{z_1}\left(\left[\sum_{j=1}^{h} e^{z_1}\right] - e^{z_1}\right)}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]} \cdot \frac{\left[\sum_{j=1}^{h} e^{z_j}\right] - e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]} \qquad = a_1(1 - a_1)$$

$$\frac{\partial a_2}{\partial z_1} = \frac{\partial}{\partial z_1}\left[\frac{e^{z_2}}{\sum_{j=1}^{h} e^{z_j}}\right]$$

	Function	Derivative
Sum Rule	$f(x) + g(x)$	$f'(x) + g'(x)$
Difference Rule	$f(x) - g(x)$	$f'(x) - g'(x)$
Product Rule	$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$
Quotient Rule	$f(x)/g(x)$	$[g(x)f'(x) - f(x)g'(x)]/[g(x)]^2$
Reciprocal Rule	$1/f(x)$	$-[f'(x)]/[f(x)]^2$
Chain Rule	$f(g(x))$	$f'(g(x))g'(x)$

$$= \frac{\left[\sum_{j=1}^{h} e^{z_j}\right]\frac{\partial}{\partial z_1}e^{z_2} - e^{z_2}\frac{\partial}{\partial z_1}\left[\sum_{j=1}^{h} e^{z_j}\right]}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{0 - e^{z_2}e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{-e^{z_2}}{\left[\sum_{j=1}^{h} e^{z_j}\right]} \cdot \frac{e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]} \qquad = -a_2a_1$$

UNIVERSIDAD EAFIT

Forma matricial

$$\nabla_{\mathbf{W}} \mathcal{L} = \mathbf{X}^\top (A - Y)$$

Base de datos balanceada:

- 10 clases
- 60.000 dígitos por clase
- Dimensión de las imágenes: $1 \times 28 \times 28$ (CHW)

La forma tradicional de abordar este problema de clasificación es convertir cada imagen en un vector de tamaño 784×1.

Si se quiere hacer entrenamiento en *batches*, cada *batch* tendría dimensión $N_b \times 784$

¿De dónde viene?

Tomemos los vectores $\mathbf{z}^{[i]} \in \Re^{K \times 1}$ y $\mathbf{y}^{[i]} \in \Re^{K \times 1}$: Probabilidad de pertenencia de la muestra i a cada una de las clases, y etiqueta de la muestra i en *one-hot-encoding*

$$l(\mathbf{y}, \mathbf{z}) = -\sum_{j=1}^{K} y_j \log \left(\frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(a_k)} \right)$$

$$l(\mathbf{y}, \mathbf{z}) = \sum_{j=1}^{K} y_j \log \sum_{k=1}^{K} \exp(z_k) - \sum_{j=1}^{K} y_j z_j$$

$$l(\mathbf{y}, \mathbf{z}) = \log \sum_{k=1}^{K} \exp(z_k) - \sum_{j=1}^{K} y_j z_j$$

Tomando la derivada con respecto a cualquier *logit* z_j

$$\frac{\partial l}{\partial z_j} = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)} - y_j$$

$$\frac{\partial l}{\partial z_j} = a_j - y_j$$

De forma vectorial:

$$\frac{\partial l}{\partial \mathbf{z}} = \mathbf{a}^{[i]} - \mathbf{y}^{[i]}$$

De forma matricial:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Z}} = \mathbf{A} - \mathbf{Y}$$

La derivada de una función lineal de la forma:

$\mathbf{Z} = \mathbf{X}\mathbf{W}^\top + \mathbf{b}$ con respecto a los parámetros $\mathbf{W}, \mathbf{b}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{X}^\top \frac{\partial \mathcal{L}}{\partial \mathbf{Z}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \frac{\partial \mathcal{L}}{\partial \mathbf{Z}}^\top \mathbf{1}_n$$

http://cs231n.stanford.edu/handouts/linear-backprop.pdf