IIC 2433 Minería de Datos

https://github.com/marcelomendoza/IIC2433

- GRADIENTE DESCENDENTE -

Regresión logística

Si el ajuste es adecuado:

$$P(y \mid \mathbf{x}) = \theta(y \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x})$$

Luego, podemos expresar la función de verosimilitud:

$$P(y_1, \ldots, y_N \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n).$$

Regresión logística

Maximizamos la función de verosimilitud:

$$
\begin{aligned}
& \max \quad \prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n) \\
\Leftrightarrow\ & \max \quad \ln\left(\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n)\right) \\
\equiv\ & \max \quad \sum_{n=1}^{N} \ln P(y_n \mid \mathbf{x}_n)
\end{aligned}
$$

Regresión logística

Maximizamos la función de verosimilitud:

$$
\begin{aligned}
& \max && \prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n) \\
\Leftrightarrow\ & \max && \ln\left(\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n)\right) \\
\equiv\ & \max && \sum_{n=1}^{N} \ln P(y_n \mid \mathbf{x}_n) \\
\Leftrightarrow\ & \min && -\frac{1}{N}\sum_{n=1}^{N} \ln P(y_n \mid \mathbf{x}_n) \\
\equiv\ & \min && \frac{1}{N}\sum_{n=1}^{N} \ln \frac{1}{P(y_n \mid \mathbf{x}_n)} \\
\equiv\ & \min && \frac{1}{N}\sum_{n=1}^{N} \ln \frac{1}{\theta(y_n \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_n)}
\end{aligned}
$$

Regresión logística

Maximizamos la función de verosimilitud:

$$
\begin{aligned}
& \max && \prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n) \\
\Leftrightarrow\ & \max && \ln\left(\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n)\right) \\
\equiv\ & \max && \sum_{n=1}^{N} \ln P(y_n \mid \mathbf{x}_n) \\
\Leftrightarrow\ & \min && -\frac{1}{N}\sum_{n=1}^{N} \ln P(y_n \mid \mathbf{x}_n) \\
\equiv\ & \min && \frac{1}{N}\sum_{n=1}^{N} \ln \frac{1}{P(y_n|\mathbf{x}_n)} \\
\equiv\ & \min && \frac{1}{N}\sum_{n=1}^{N} \ln \frac{1}{\theta(y_n \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_n)} \\
\equiv\ & \min && \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_n}\right)
\end{aligned}
$$

$$\theta(s) = \frac{1}{1+e^{-s}}.$$

Regresión logística

Tenemos una expresión para:

Parámetros del modelo

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_n})$$

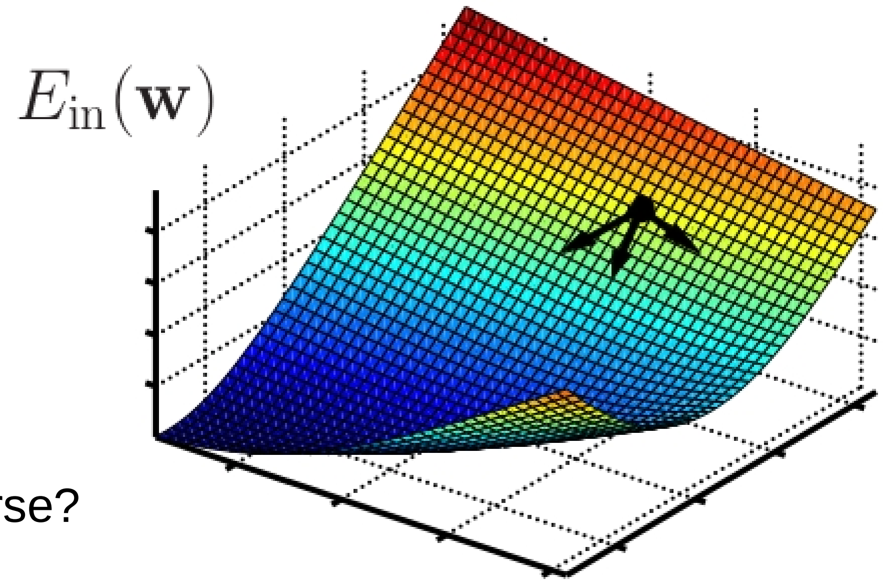
Cross-entropy

La función es convexa, por lo que podemos optimizarla de forma iterativa:

Idea del gradiente descendente:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\hat{\mathbf{v}}$$

¿En cuál dirección conviene moverse?



Gradiente descendente

$$\begin{aligned}\Delta E_{\text{in}} &= E_{\text{in}}(\mathbf{w}(t+1)) - E_{\text{in}}(\mathbf{w}(t)) \\ &= E_{\text{in}}(\mathbf{w}(t) + \eta\hat{\mathbf{v}}) - E_{\text{in}}(\mathbf{w}(t)) \\ &= \eta\, \nabla E_{\text{in}}(\mathbf{w}(t))^{\mathrm{T}}\hat{\mathbf{v}} + O(\eta^2)\end{aligned}$$

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(k+1)}(c)}{(k+1)!}(x - x_0)^{k+1}.$$

$$\Delta E_{\text{in}} = E_{\text{in}}(\mathbf{w}(t+1)) - E_{\text{in}}(\mathbf{w}(t))$$

$$\underbrace{f(x_0 + \eta \cdot x)}_{f(x_0) + \nabla f(x_0) \cdot (x_0 + \eta \cdot x - x_0) + \ldots} - f(x_0)$$

(expansión de Taylor de 1er orden)

Gradiente descendente

$$\begin{aligned}\Delta E_{\text{in}} &= E_{\text{in}}(\mathbf{w}(t+1)) - E_{\text{in}}(\mathbf{w}(t)) \\ &= E_{\text{in}}(\mathbf{w}(t)+\eta\hat{\mathbf{v}}) - E_{\text{in}}(\mathbf{w}(t)) \\ &= \eta\,\nabla E_{\text{in}}(\mathbf{w}(t))^{\mathrm{T}}\hat{\mathbf{v}} + O(\eta^2)\end{aligned}$$

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(k+1)}(c)}{(k+1)!}(x-x_0)^{k+1}.$$

$$\Delta E_{\text{in}} = E_{\text{in}}(\mathbf{w}(t+1)) - E_{\text{in}}(\mathbf{w}(t))$$

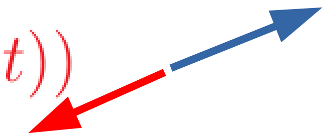
$$\underbrace{f(x_0+\eta\cdot x)}_{f(\cancel{x_0}) + \nabla f(x_0)\cdot(\cancel{x_0} + \eta\cdot x - \cancel{x_0}) + \dots} - f(\cancel{x_0})$$

(expansión de Taylor de 1er orden)

Gradiente descendente

$$-\nabla E_{\text{in}}(\mathbf{w}(t))$$

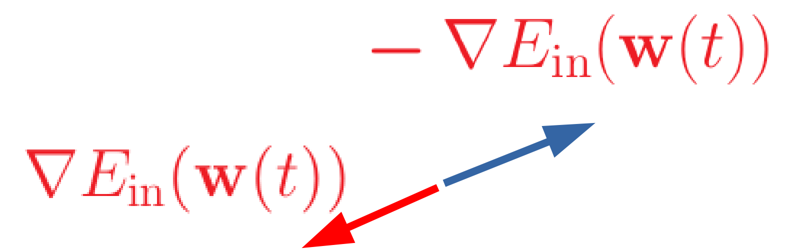
$$= \ \eta \underbrace{\nabla E_{\text{in}}(\mathbf{w}(t))^{\mathrm{T}}\hat{\mathbf{v}}} + O(\eta^2) \qquad \nabla E_{\text{in}}(\mathbf{w}(t))$$



Minimizado en $\hat{\mathbf{v}} = -\frac{\nabla E_{\text{in}}(\mathbf{w}(t))}{\| \nabla E_{\text{in}}(\mathbf{w}(t)) \|}$

Gradiente descendente

$$= \eta \underbrace{\nabla E_{\text{in}}(\mathbf{w}(t))^{\mathrm{T}}\hat{\mathbf{v}}} + O(\eta^2)$$



Minimizado en $\hat{\mathbf{v}} = -\dfrac{\nabla E_{\text{in}}(\mathbf{w}(t))}{\|\nabla E_{\text{in}}(\mathbf{w}(t))\|}$

Gradiente descendente

Initialize at step $t = 0$ to $\mathbf{w}(0)$.
for $t = 0, 1, 2, \ldots$ **do**
  Compute the gradient

$$\mathbf{g}_t = \nabla E_{\text{in}}(\mathbf{w}(t)).$$

  Move in the direction $\mathbf{v}_t = -\mathbf{g}_t$.
  Update the weights:

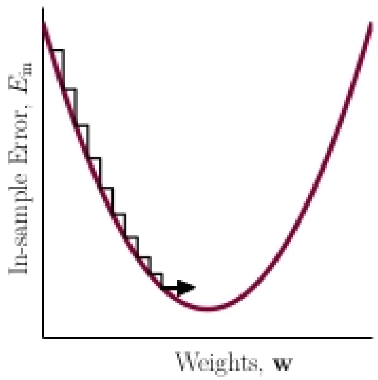
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\mathbf{v}_t.$$

  Iterate 'until it is time to stop'.
end for
Return the final weights.

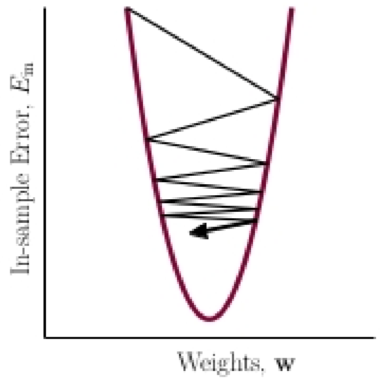
Gradiente descendente

El efecto del **learning rate**:

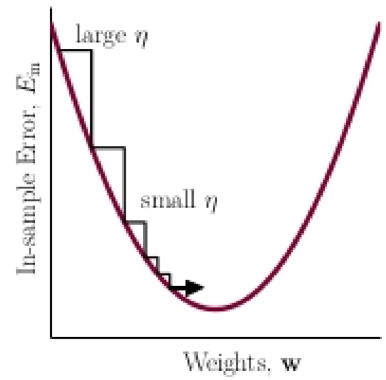
η muy pequeño



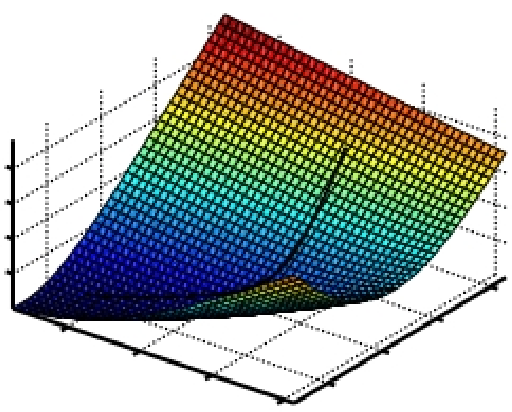
η muy grande



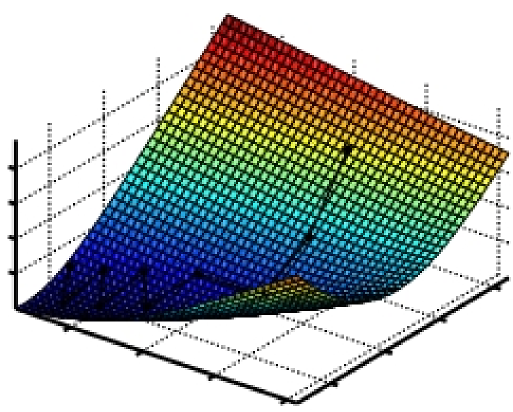
η_t



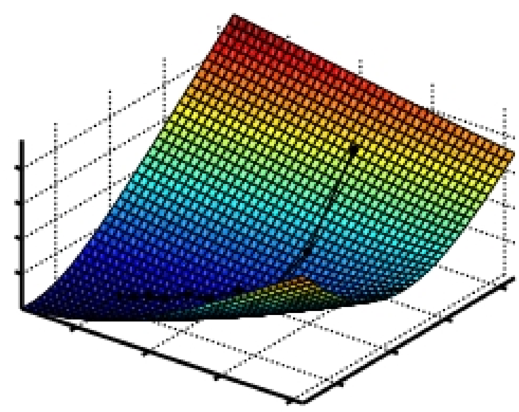
Adam



$\eta = 0.1$; 75 steps



$\eta = 2$; 10 steps



variable η_t; 10 steps

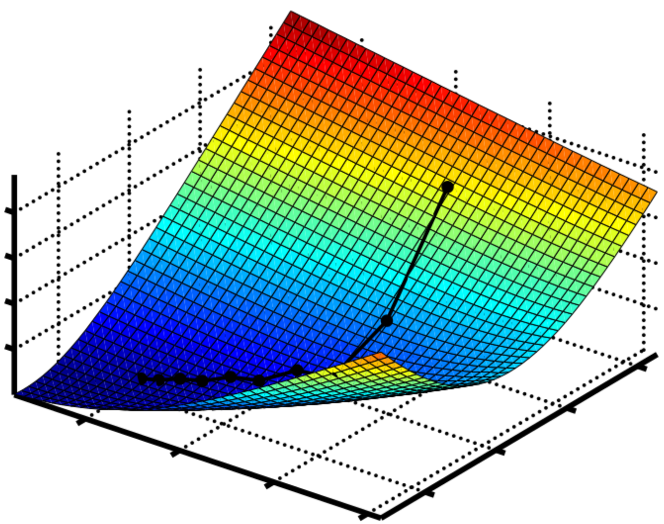
Gradiente descendente

Una variación de gradiente descendente considera la evaluación del gradiente en una muestra del training set.

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \eta \nabla_{\mathbf{w}} e(\mathbf{w}, \mathbf{x}_*, y_*)$$

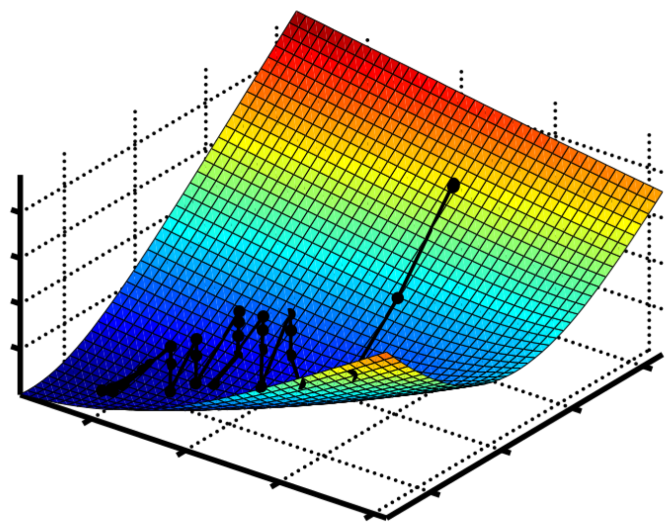
Dado que la muestra se toma al azar, se le denomina gradiente descendente estocástico.

GD



$\eta = 6$
10 steps
$N = 10$

SGD



$\eta = 2$
30 steps

Puede ayudar a escapar de óptimos locales

Regresión logística + gradiente descendente

Para regresión logística encontramos que:

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_n})$$

Muestre que:

$$\begin{aligned} \nabla E_{\text{in}}(\mathbf{w}) &= -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^{\mathrm{T}}\mathbf{x}_n}} \\ &= \frac{1}{N}\sum_{n=1}^{N} -y_n \mathbf{x}_n \theta(-y_n \mathbf{w}^{\mathrm{T}}\mathbf{x}_n). \end{aligned}$$

Regresión logística + gradiente descendente

Logistic regression algorithm:

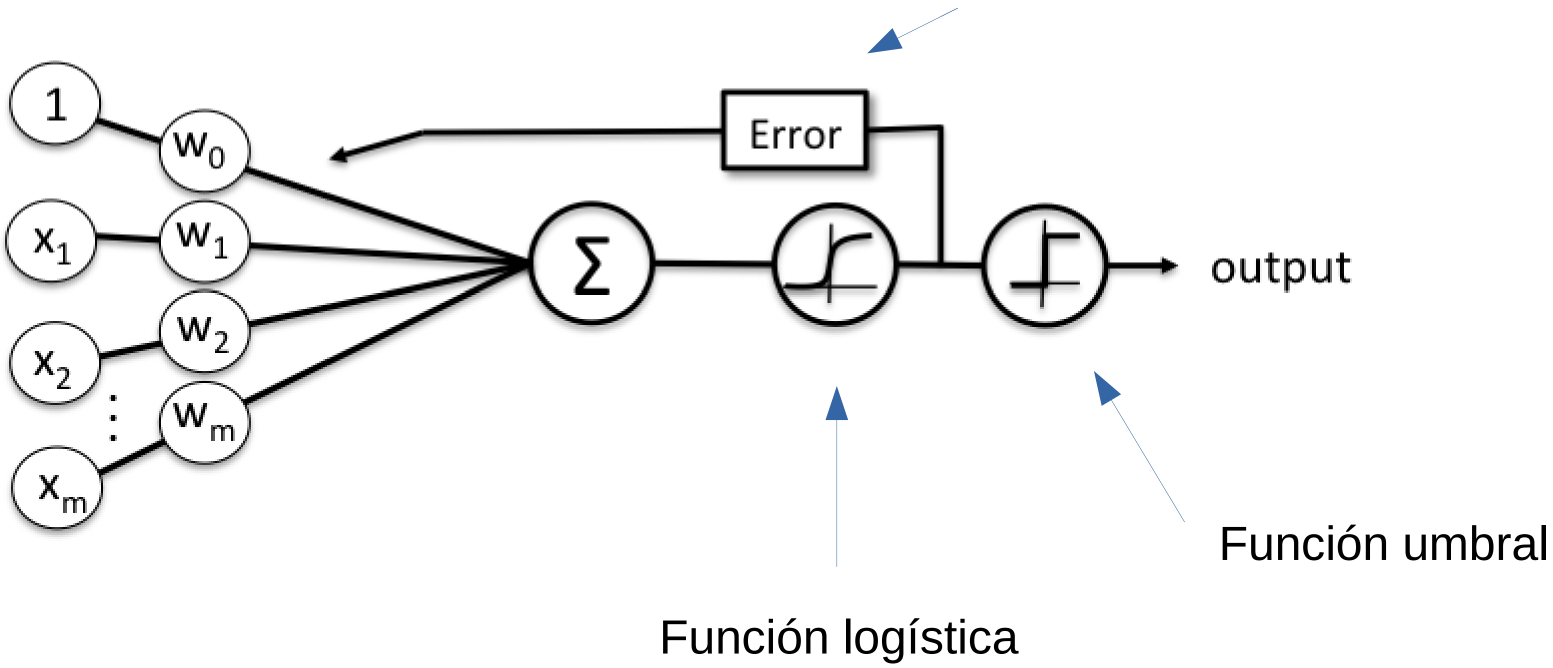
1: Initialize the weights at time step $t = 0$ to $\mathbf{w}(0)$.
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Compute the gradient

$$\mathbf{g}_t = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\mathbf{x}_n}{1+e^{y_n\mathbf{w}^{\mathrm{T}}(t)\mathbf{x}_n}}.$$

4: Set the direction to move, $\mathbf{v}_t = -\mathbf{g}_t$.
5: Update the weights: $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\mathbf{v}_t$.
6: Iterate to the next step until it is time to stop.
7: Return the final weights $\mathbf{w}$.

Regresión logística + gradiente descendente

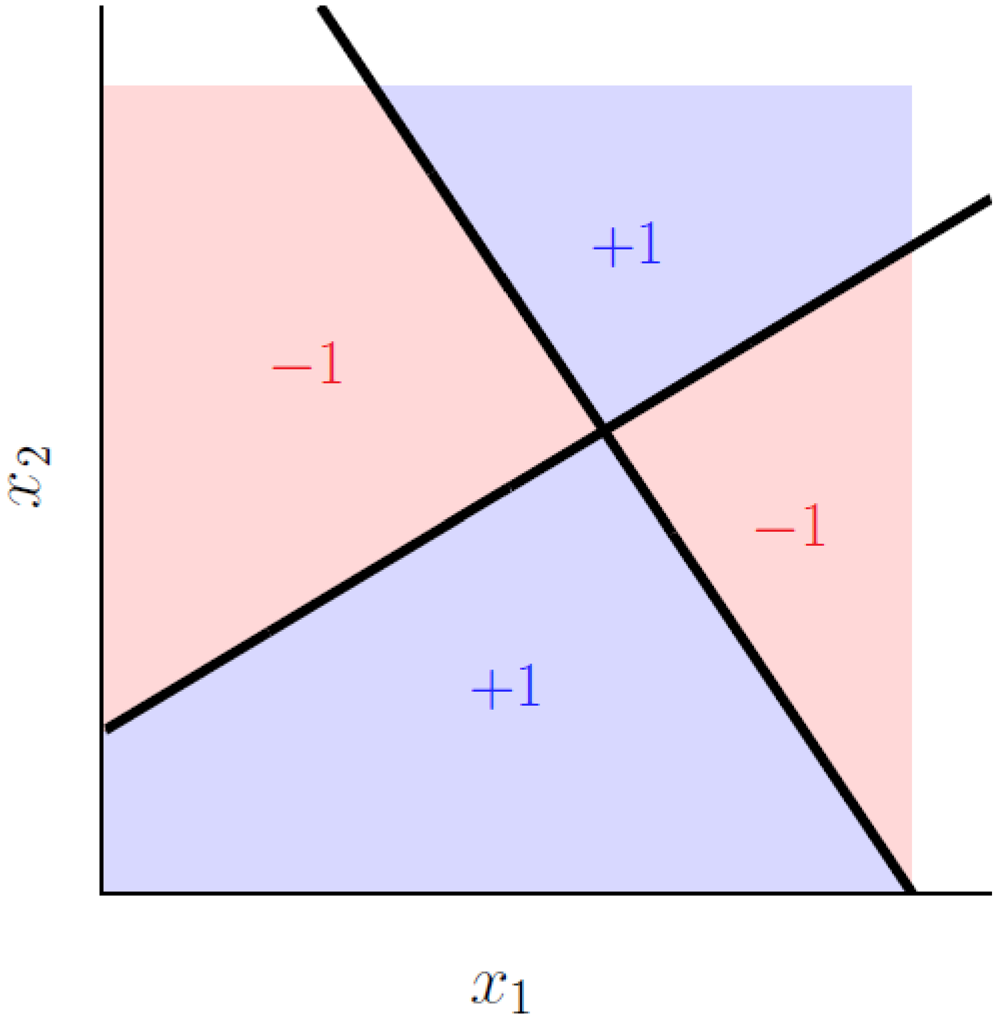
$$\approx \quad -\eta \| \nabla E_{\text{in}}(\mathbf{w}(t)) \|$$



- MLP -

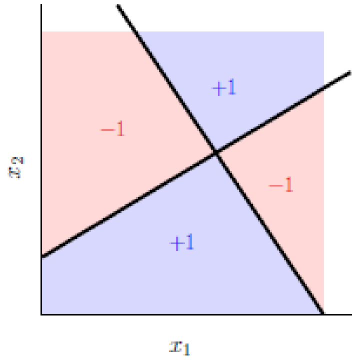
La limitación del modelo lineal

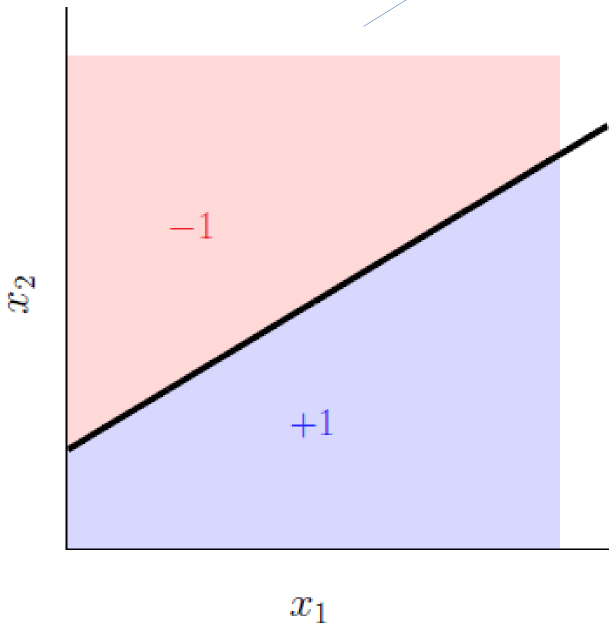
XOR:



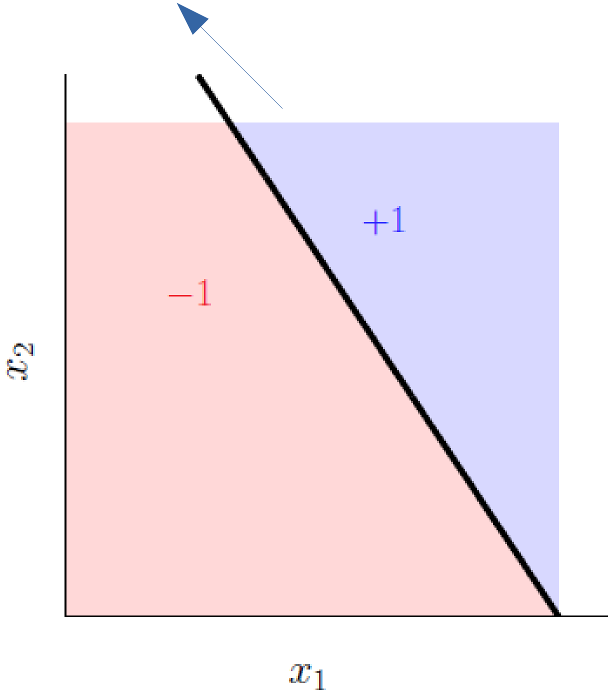
Combinación de perceptrones

XOR:





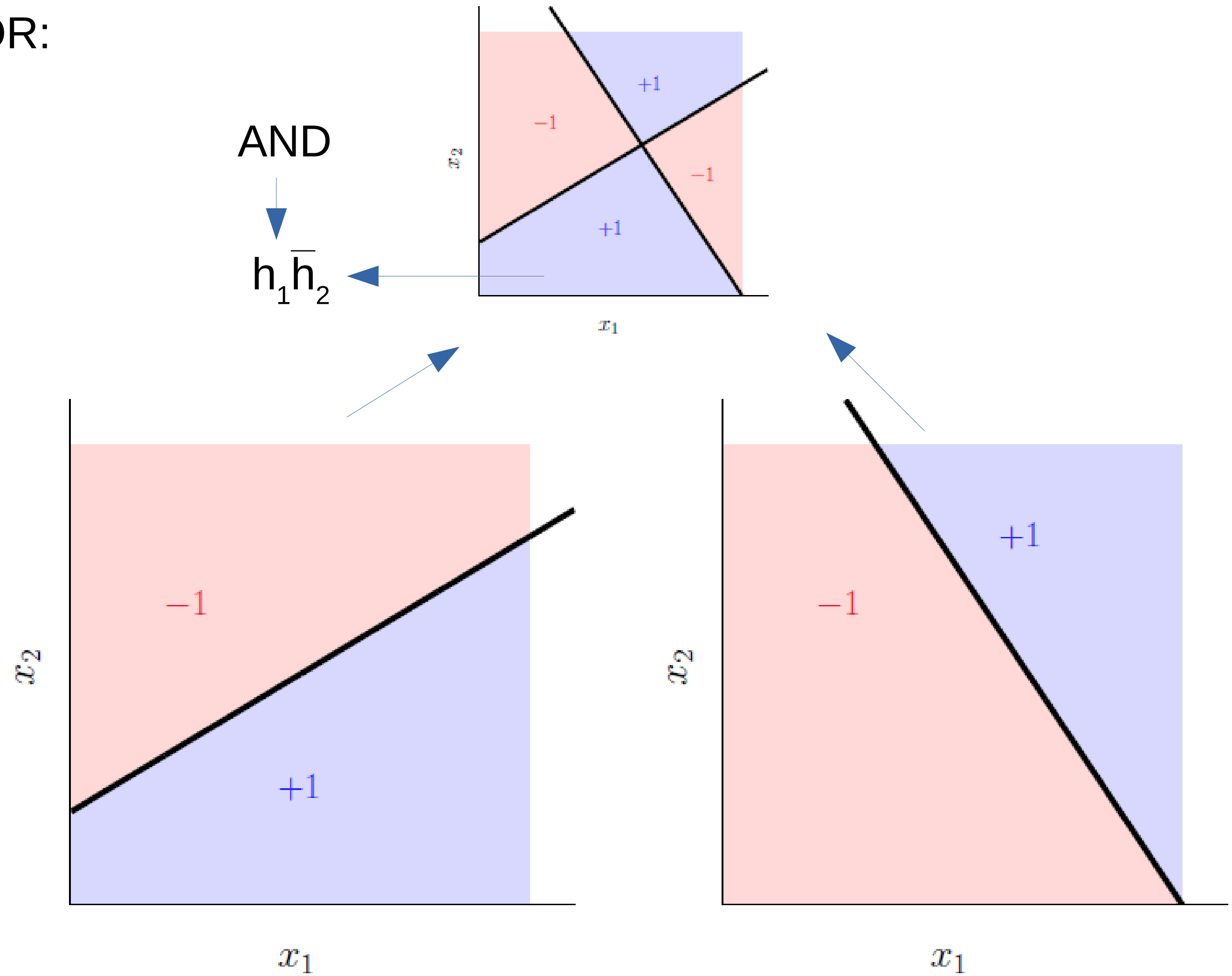
$h_1(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_1^{\mathrm{T}}\mathbf{x})$



$h_2(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_2^{\mathrm{T}}\mathbf{x})$

Combinación de perceptrones

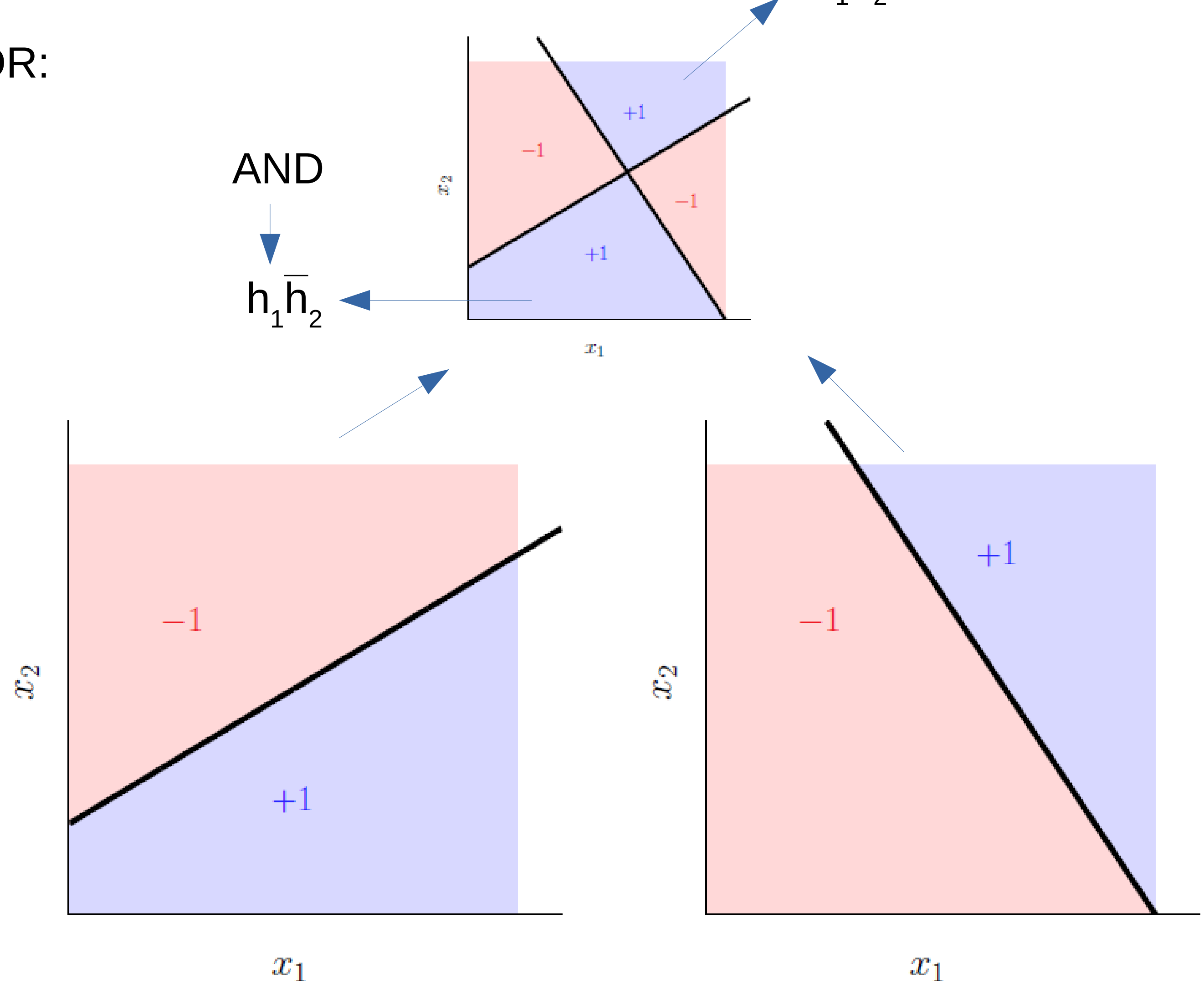
XOR:



$$h_1(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_1^{\mathrm{T}}\mathbf{x}) \qquad h_2(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_2^{\mathrm{T}}\mathbf{x})$$

Combinación de perceptrones

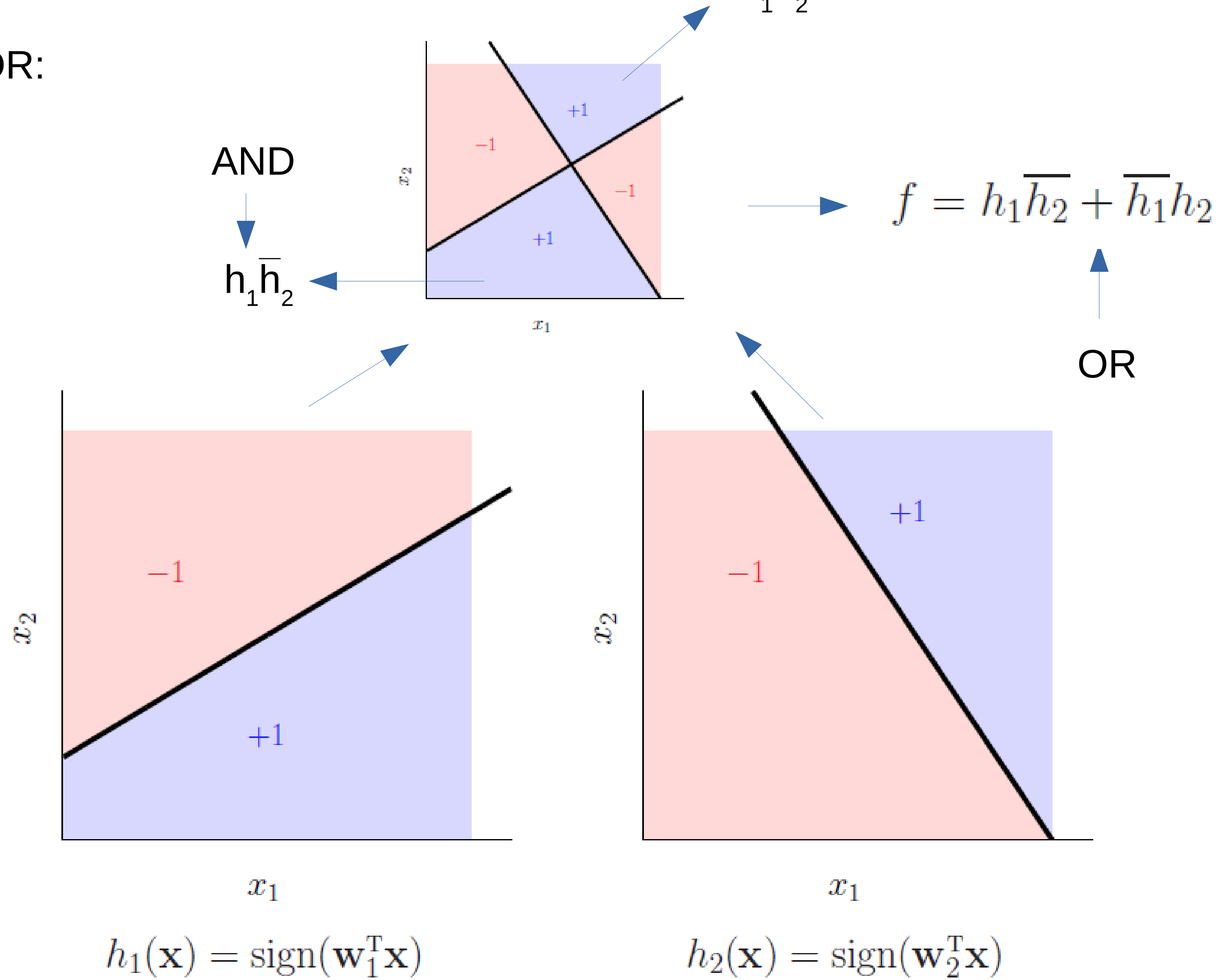
XOR:



$h_1(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_1^{\mathrm{T}}\mathbf{x})$ $\qquad$ $h_2(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_2^{\mathrm{T}}\mathbf{x})$

Combinación de perceptrones

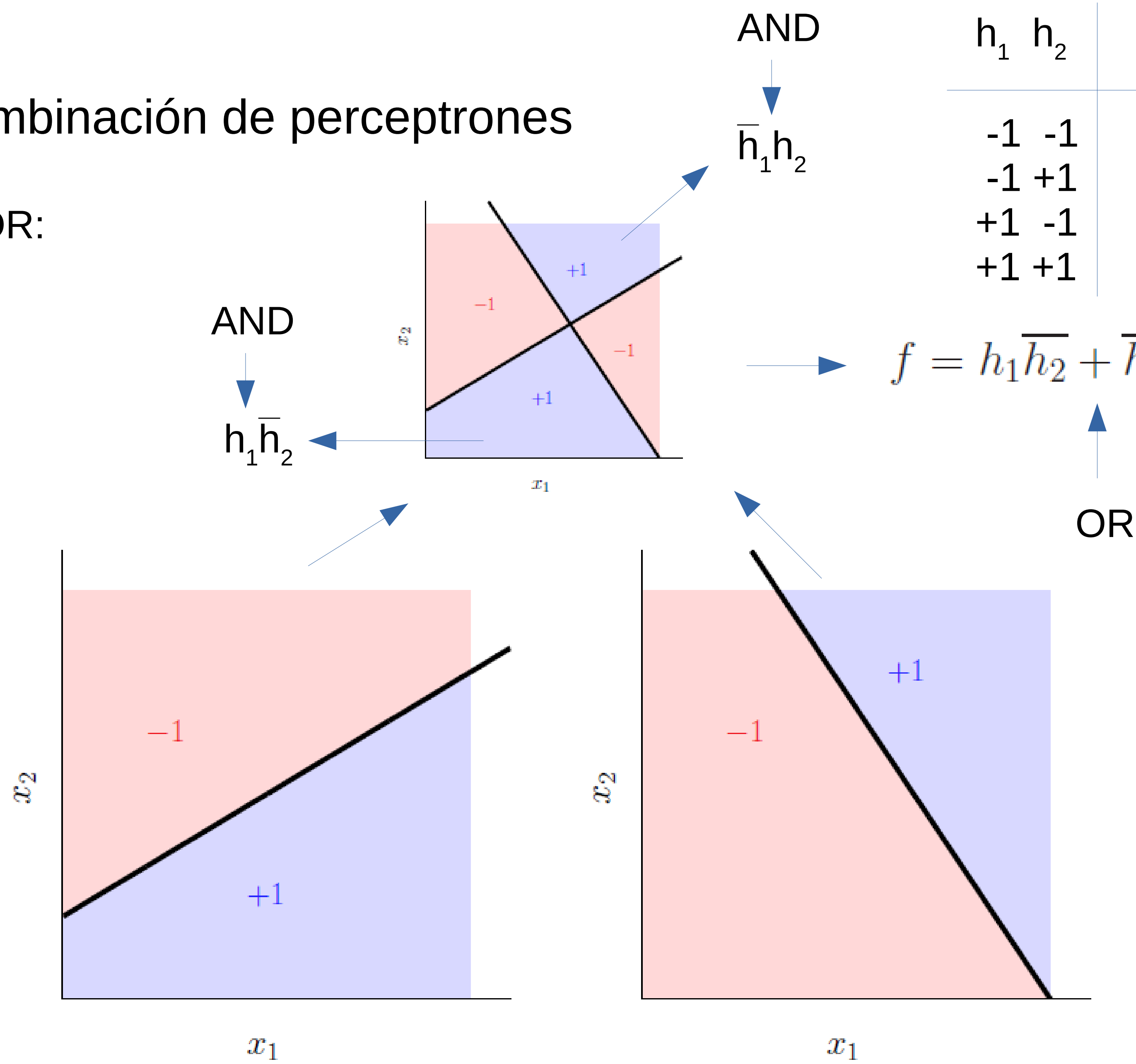
XOR:



Combinación de perceptrones

XOR:

h_1 h_2	XOR
-1 -1	-1
-1 +1	+1
+1 -1	+1
+1 +1	-1



$$f = h_1\overline{h_2} + \overline{h_1}h_2$$

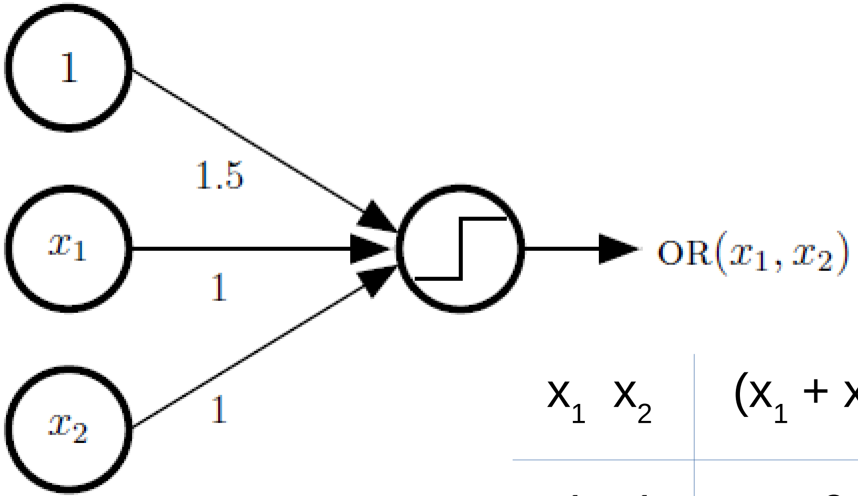
$h_1(\mathbf{x}) = \text{sign}(\mathbf{w}_1^{\mathrm{T}}\mathbf{x})$

$h_2(\mathbf{x}) = \text{sign}(\mathbf{w}_2^{\mathrm{T}}\mathbf{x})$

Combinación de perceptrones

OR:

$$\mathrm{OR}(x_1, x_2) = \mathrm{sign}(x_1 + x_2 + 1.5)$$

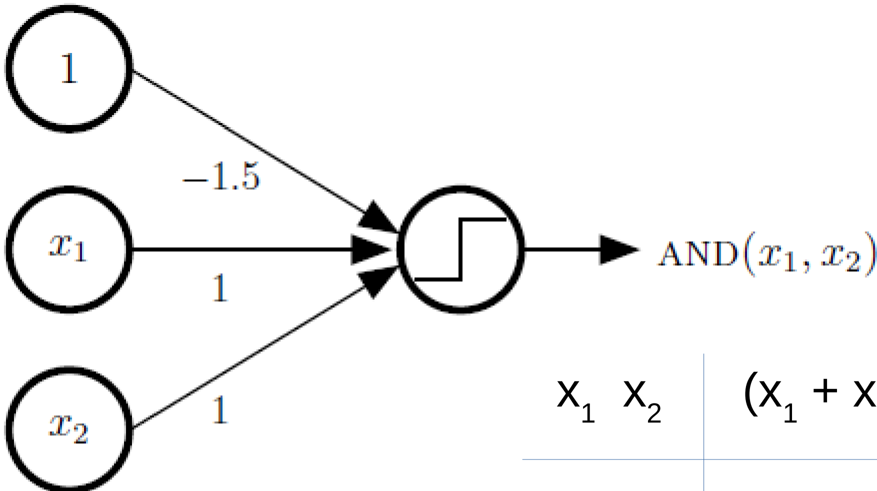


x_1 x_2	$(x_1 + x_2 + 1.5)$	signo
-1 -1	-0.5	-1
-1 +1	1.5	+1
+1 -1	1.5	+1
+1 +1	3.5	+1

Combinación de perceptrones

AND:

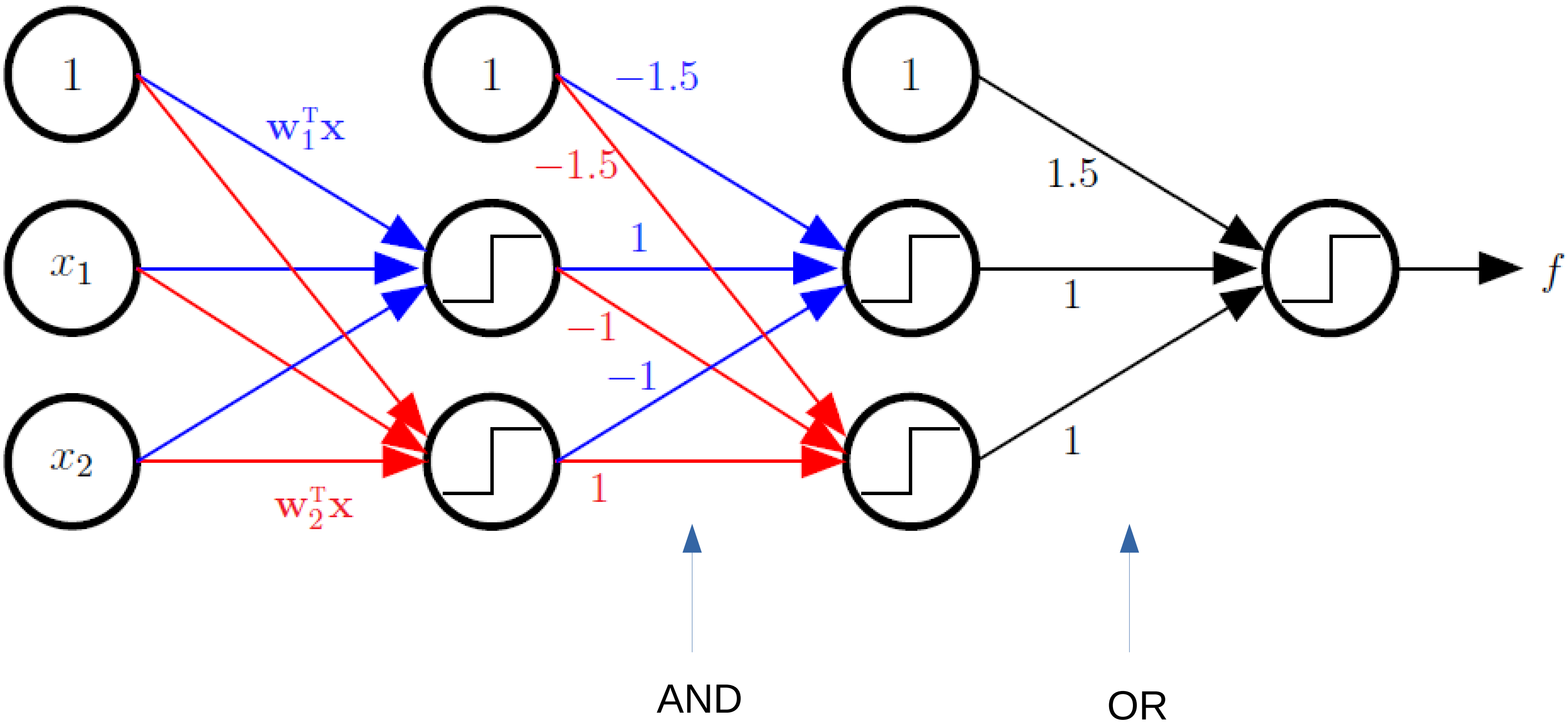
$$\mathrm{AND}(x_1, x_2) = \mathrm{sign}(x_1 + x_2 - 1.5)$$



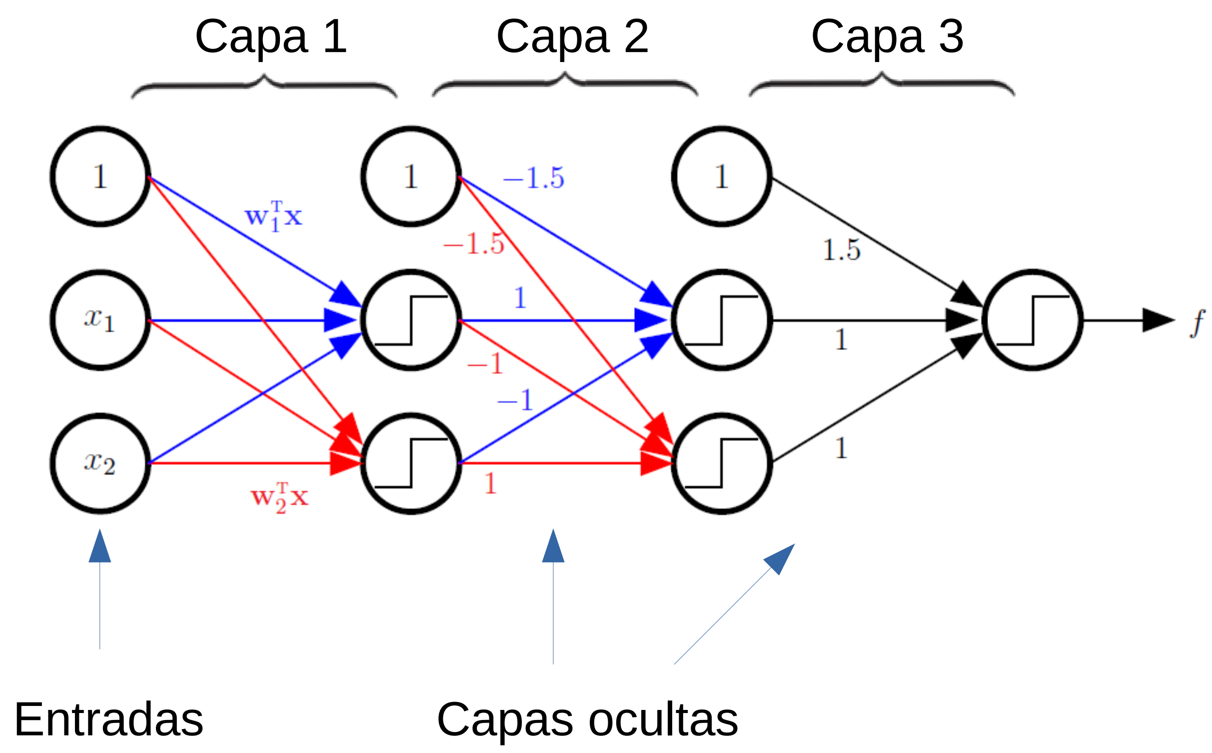
x_1 x_2	$(x_1 + x_2 - 1.5)$	signo
-1 -1	-3.5	-1
-1 +1	-1.5	-1
+1 -1	-1.5	-1
+1 +1	0.5	+1

Combinación de perceptrones

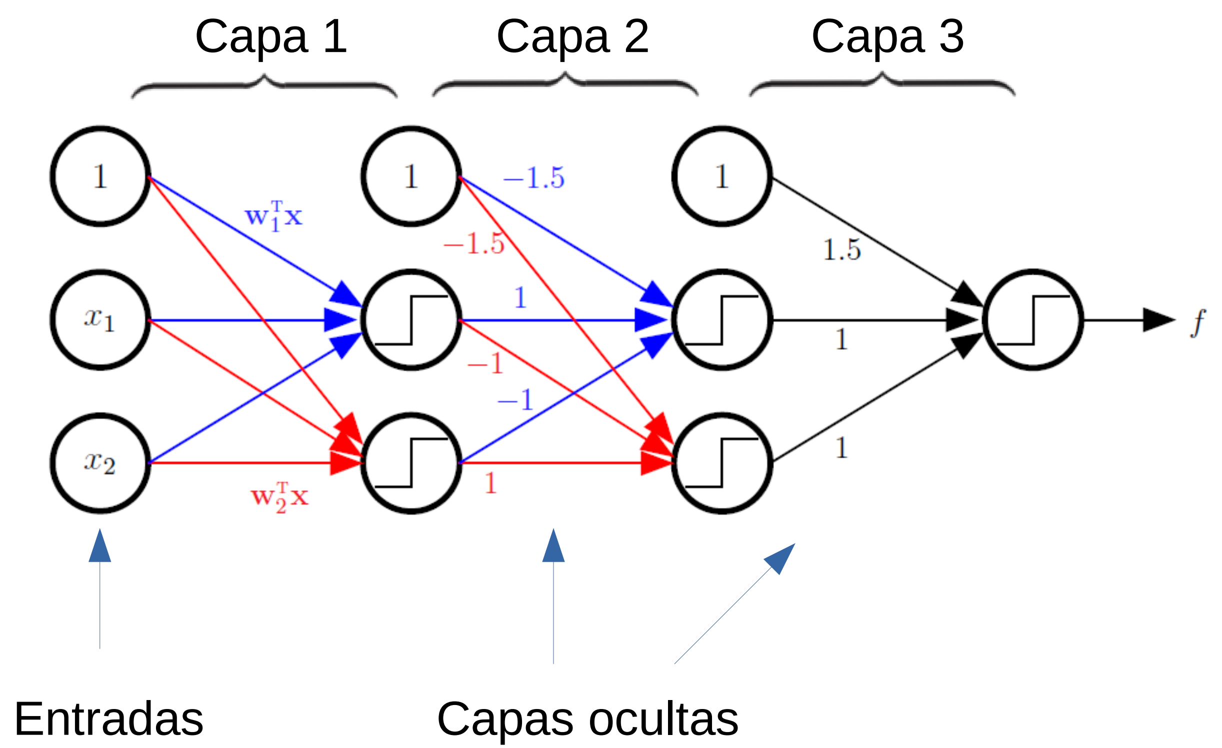
$$f = h_1\overline{h_2} + \overline{h_1}h_2$$



El perceptrón multicapa (Multi-Layer Perceptron)



El perceptrón multicapa (Multi-Layer Perceptron)

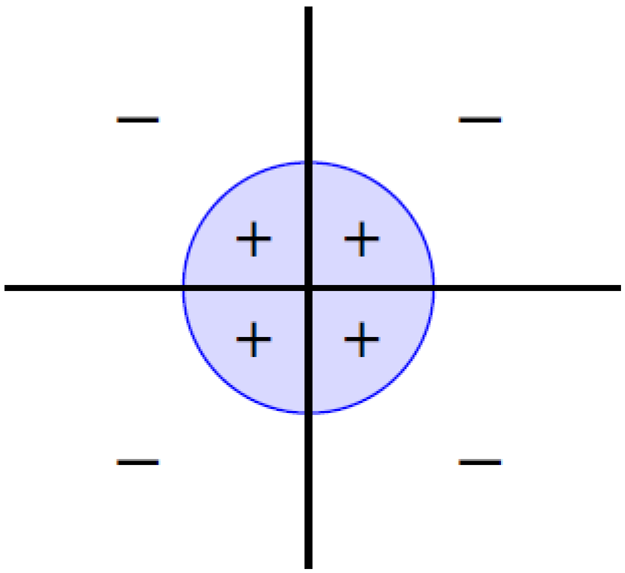


Aproximación universal

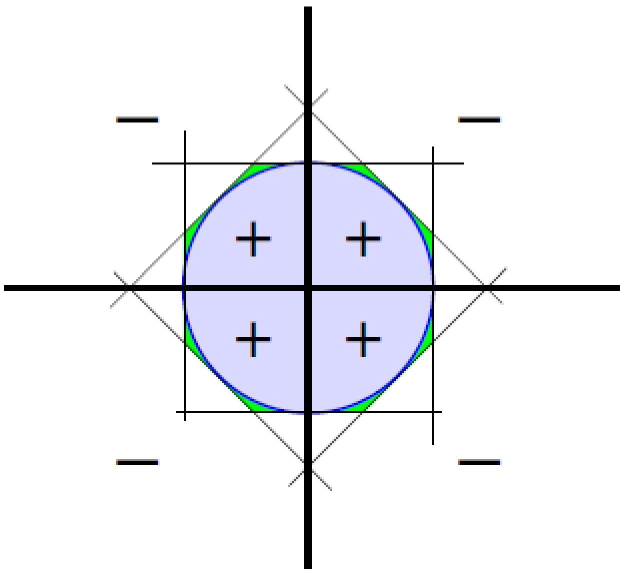
Cualquier función que se puede descomponer en separadores lineales puede ser implementada por un MLP de 3 capas

El perceptrón multicapa (Multi-Layer Perceptron)

Un separador suave puede ser aproximado por N separadores lineales.



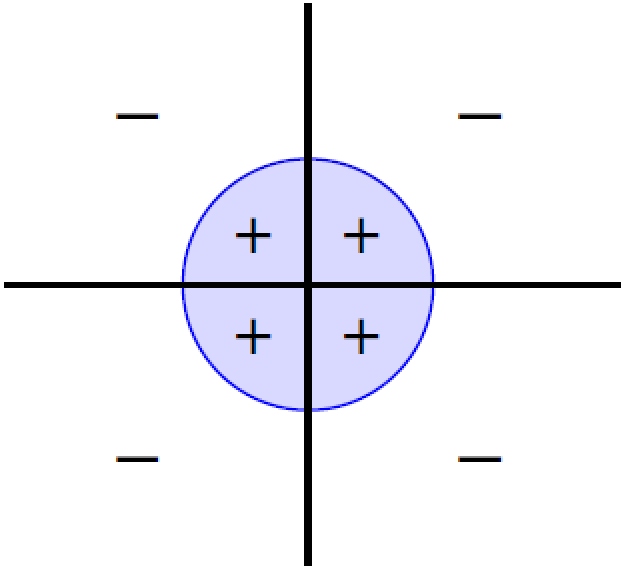
f

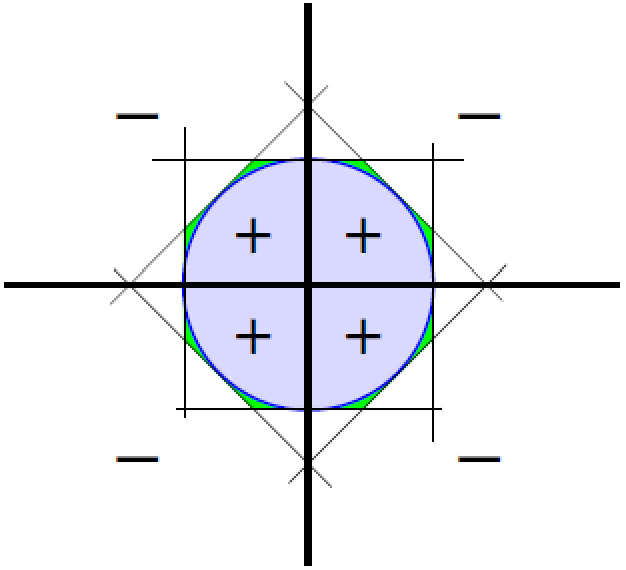


¿Cuántos perceptrones hay?

El perceptrón multicapa (Multi-Layer Perceptron)

Un separador suave puede ser aproximado por N separadores lineales.

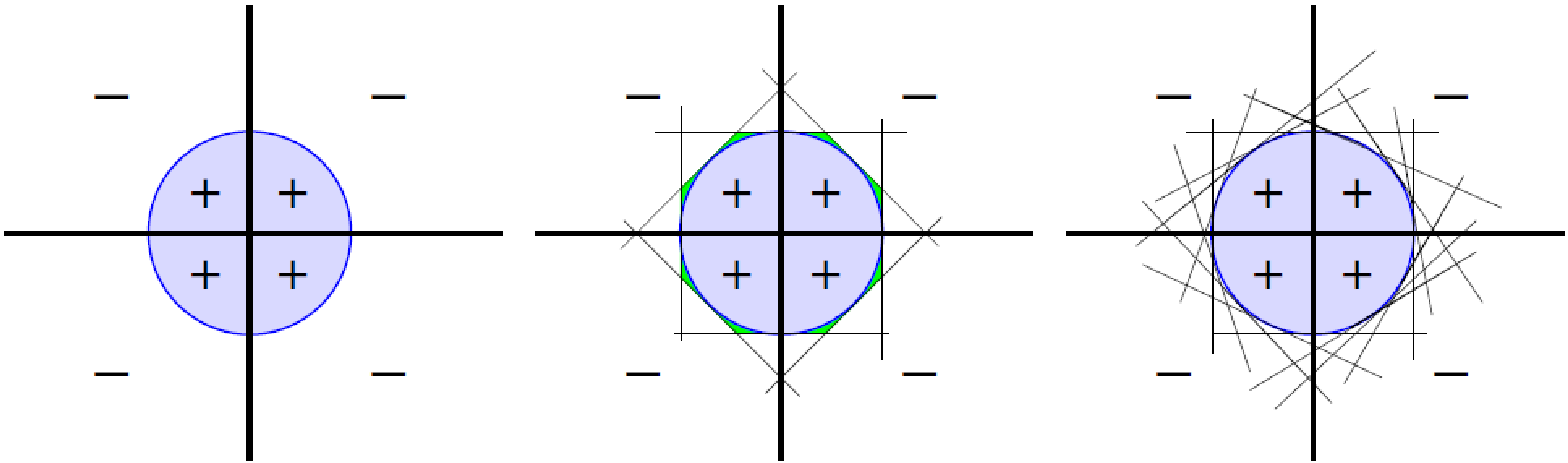




f 8

El perceptrón multicapa (Multi-Layer Perceptron)

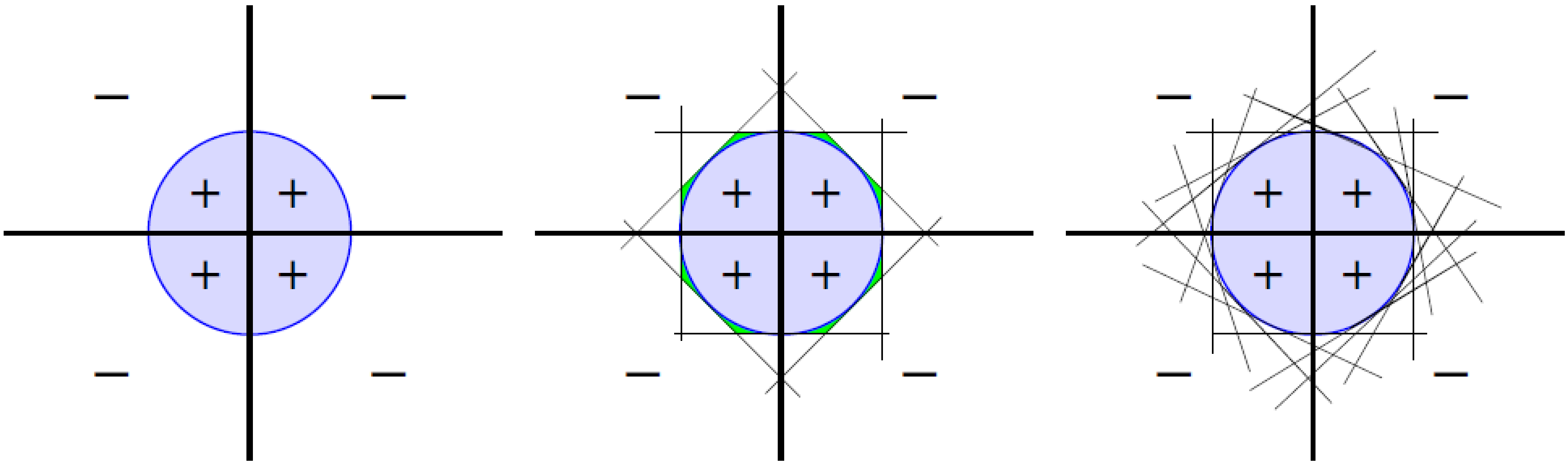
Un separador suave puede ser aproximado por N separadores lineales.



f 8 ¿?

El perceptrón multicapa (Multi-Layer Perceptron)

Un separador suave puede ser aproximado por N separadores lineales.



f 8 ¿?

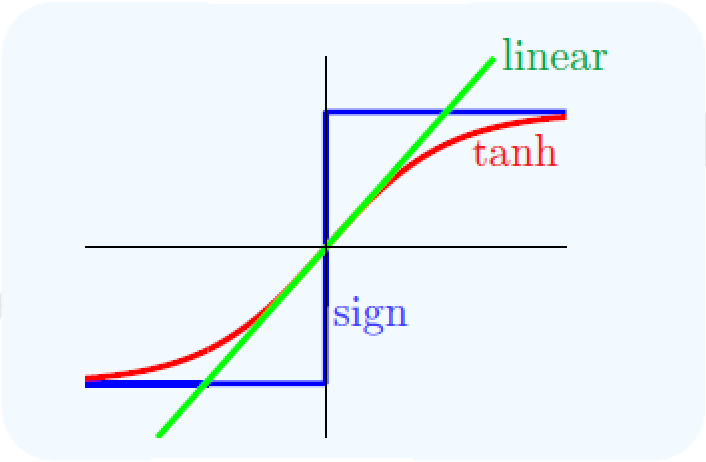
Tradeoff aproximación-generalización:

Más neuronas, mejor aproximación.
Más neuronas, peor generalización.

El perceptrón multicapa (Multi-Layer Perceptron)

Para entrenar un MLP necesitamos reemplazar la función signo dado que no es diferenciable.

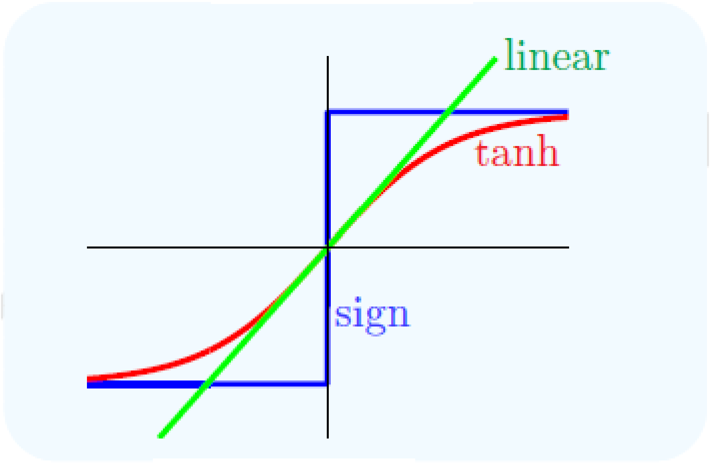
Podemos suavizar la función signo con la tangente hiperbólica, que tiene un comportamiento lineal cerca del origen y es cercana a +1 o -1 para entradas grandes.

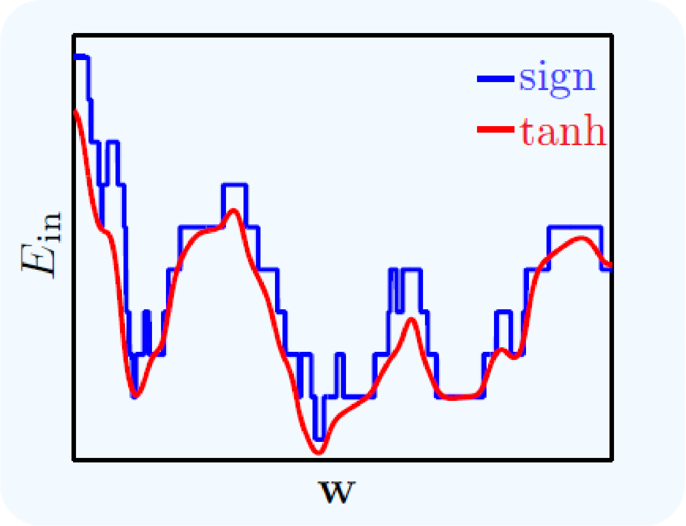


El perceptrón multicapa (Multi-Layer Perceptron)

Para entrenar un MLP necesitamos reemplazar la función signo dado que no es diferenciable.

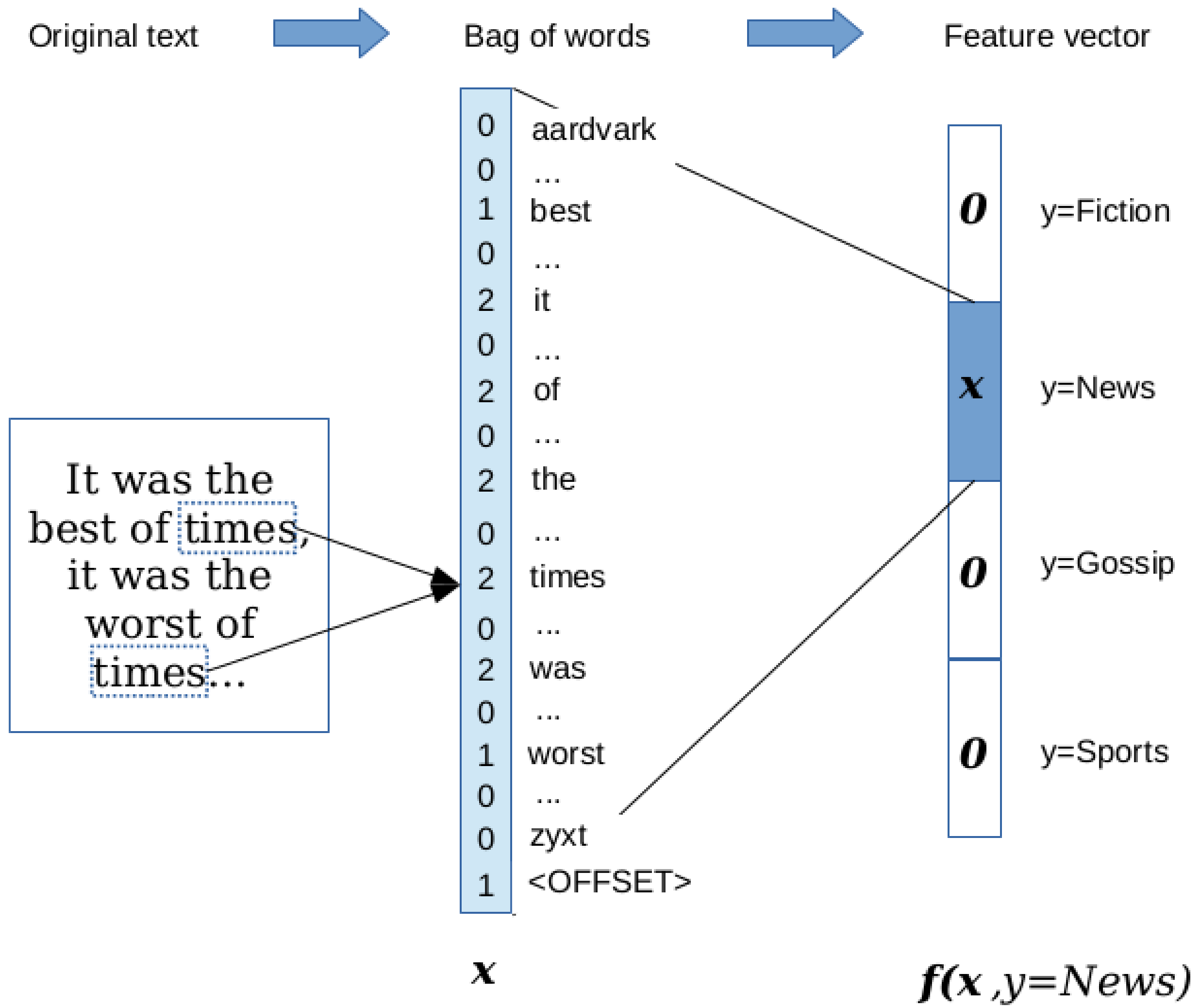
Podemos suavizar la función signo con la tangente hiperbólica, que tiene un comportamiento lineal cerca del origen y es cercana a +1 o -1 para entradas grandes.





- CLASIFICACIÓN DE TEXTO -

BOW



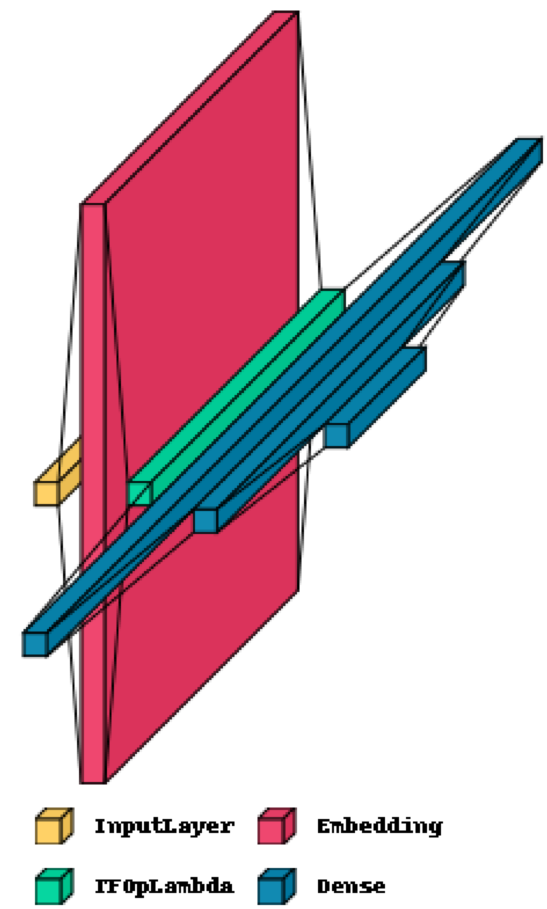
Original text
Bag of words
Feature vector
It was the best of times, it was the worst of times...
0 aardvark
0 ...
1 best
0 ...
2 it
0 ...
2 of
0 ...
2 the
0 ...
2 times
0 ...
2 was
0 ...
1 worst
0 ...
0 zyxt
1 <OFFSET>
x
0 y=Fiction
x y=News
0 y=Gossip
0 y=Sports
f(x ,y=News)

Clasificación con word embeddings

|V|

```
inputs = Input(shape=(max_tokens, ))

embeddings_layer = Embedding(input_dim=len(tokenizer.index_word)+1,
output_dim=embed_len, input_length=max_tokens, trainable=True)

dense1 = Dense(128, activation="relu")
dense2 = Dense(64, activation="relu")
dense3 = Dense(len(classes), activation="softmax")

x = embeddings_layer(inputs)
x = tensorflow.reduce_sum(x, axis=1)
x = dense1(x)
x = dense2(x)
outputs = dense3(x)

model = Model(inputs=inputs, outputs=outputs)
```

forward



Clasificación con word embeddings