Grado en Ingeniería Informática
Departamento de Ingeniería Informática
Universidad de Cádiz

Tema 7
REDES NEURONALES ARTIFICIALES.
EL PERCEPTRON

Objetivos

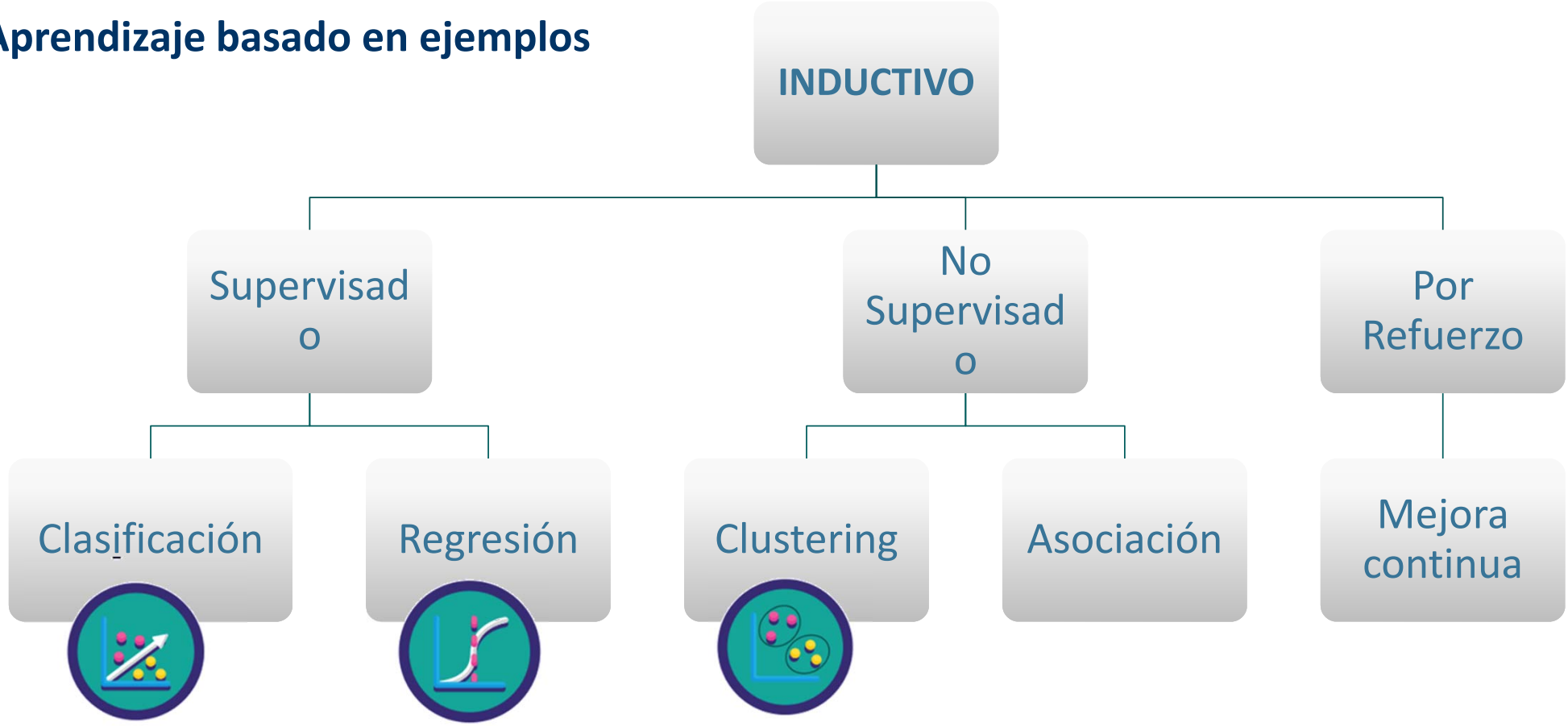
Al finalizar este tema el estudiante debe ser capaz de:

- Conocer y diferenciar claramente entre aprendizaje supervisado, no supervisado y por refuerzo, así como las tareas específicas que pueden resolverse.
- Conocer la estructura básica y la regla de aprendizaje del Perceptrón monocapa, para resolver problemas linealmente separables.
- Conocer la arquitectura general de un Perceptrón Multicapa, las fases y el proceso de entrenamiento.
- Implementar ambas redes y saber modificar los parámetros relacionados con la arquitectura, las funciones de activación, etcétera.
- Aplicar medidas para el análisis del rendimiento del sistema y de su capacidad de generalización.
- Saber analizar los resultados y realizar mejoras sobre los mismos.

Contenidos

1. Introducción
Aprendizaje INDUCTIVO

Aprendizaje basado en ejemplos



1. Introducción
Fases en el Aprendizaje Supervisado

Fase de **aprendizaje (entrenamiento)**: la red se entrena para que responda de acuerdo a los datos de entrenamiento facilitados a la red:

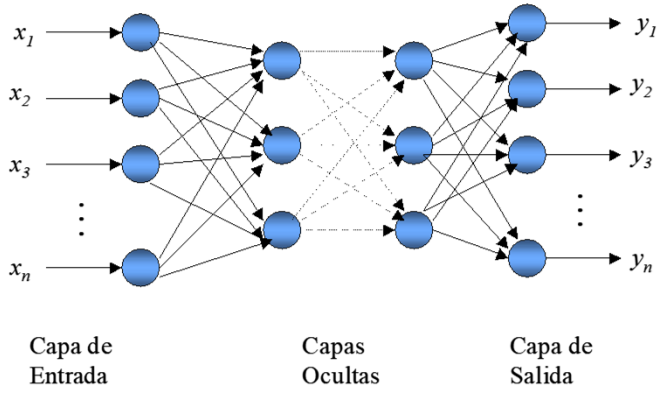
Se usan datos de entrenamiento donde se conoce la entrada y la salida (datos etiquetados)	Se modifican los pesos de las conexiones.

Fase de **presentación (generalización):** la red, ya entrenada, se utiliza para realizar las predicciones sobre datos nuevos:

Los datos son nuevos, su salida no es conocida a priori.	Los pesos ya no se modifican.

2. La neurona artificial
Modelo neuronal de McCulloch-Pitts

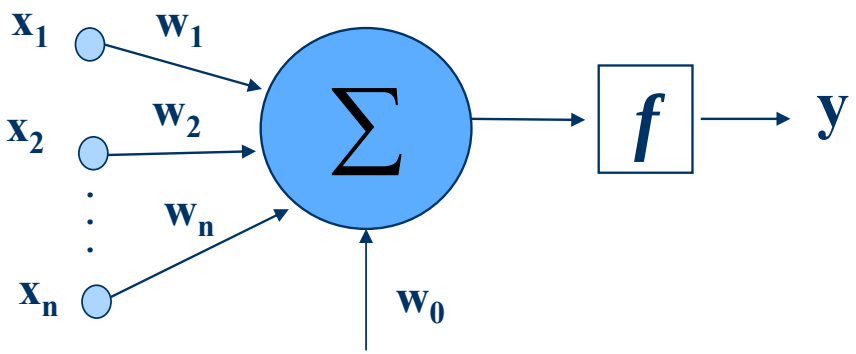
- Red de elementos simples (usualmente adaptativos) y con organización jerárquica que se interconectan masivamente en paralelo.
  - Intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico [Kohonen, 88]



2. La neurona artificial

Una neurona artificial consta de los siguientes elementos:

- Un conjunto de entradas X={ $x1,\ldots x_n$ }.
- Los pesos sinápticos $w1,\ldots w_n$, correspondientes a cada entrada.
- Un peso adicional, w_0.
- Una función de agregación, Σ.
- Una función de activación, f.
- Una salida y.



2. La neurona artificial

- **Entrada:** Cada neurona recibe diferentes señales de entrada (x_1, x_2, ...) de fuentes externas o de otras neuronas.
- **Pesos:** Cada señal de entrada se multiplica por un valor, denominado peso (w_1, w_2, ...):

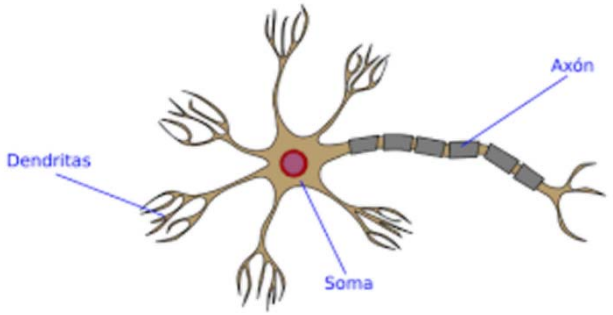
$$(x_i * w_i)$$

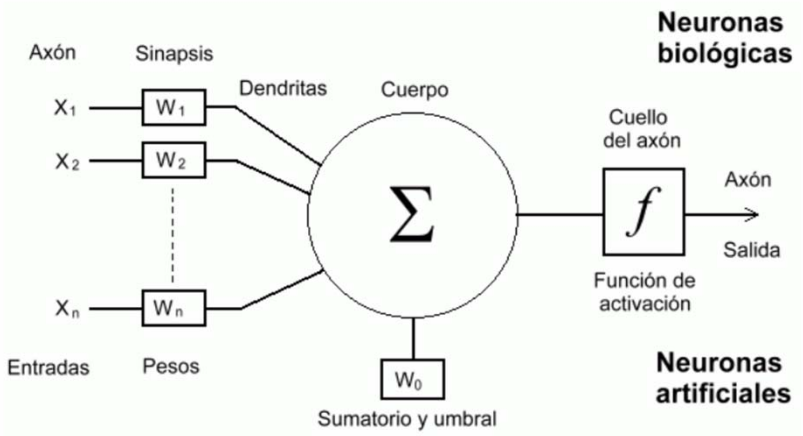
- **Entrada Neta $\sum$:** se realiza la suma de las entradas ponderadas por sus pesos

$$Neta = \sum(x_1 * w_1, \ldots x_n * w_n)$$

- **Salida o Nivel de Activación**: estado interno de la neurona, aplicando una función de activación(**f**) a la entrada neta:

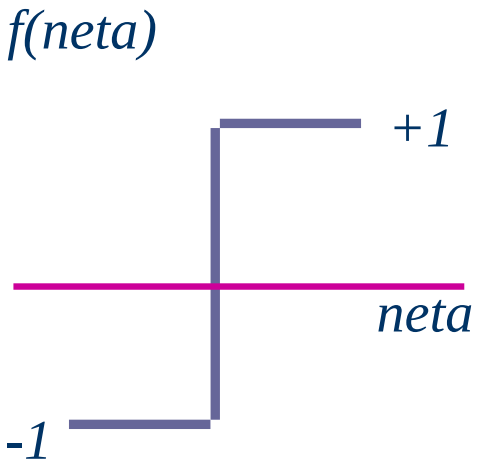
$$Y = f(neta)$$

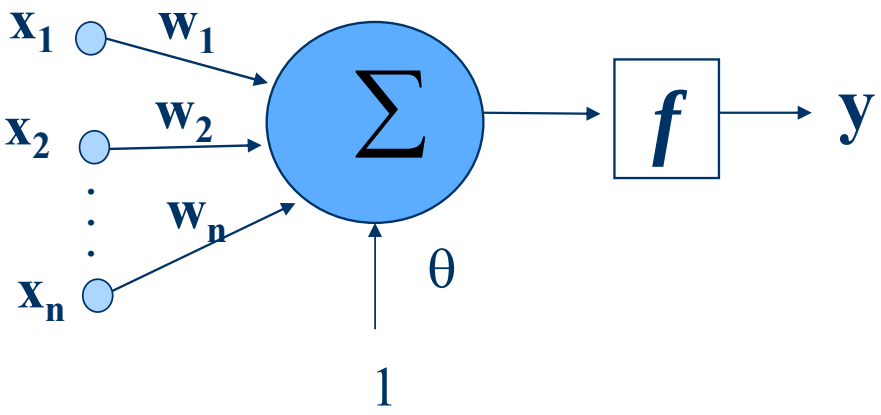




2.1 El PERCEPTRON (McCulloch-Pitts)

- La función de activación es la función umbral.





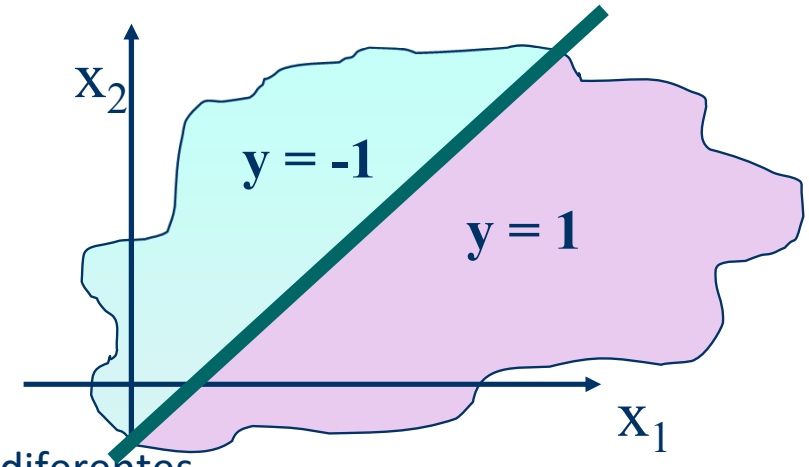
$$y = f_\theta(neta) = \begin{cases} 1 \;\; si \quad neta = \sum_i x_i * w_i > \theta \\ -1 \;\; si \quad neta = \sum_i x_i * w_i <= \theta \end{cases}$$

2.1 El PERCEPTRON
Eliminación del umbral

- La salida de la unidad es:
  - si $w_1 * x_1 + w_2 * x_2 - \theta > 0$, entonces y= 1
  - si $w_1 * x_1 + w_2 * x_2 - \theta <= 0$, entonces y= -1

- Interpretaremos la ecuación como la función de una recta:
  - $w_1 * x_1 + w_2 * x_2 - \theta = 0$

- La recta se utiliza como función discriminante entre dos clases diferentes.



2.2 Algoritmo del perceptrón
Deducción intuitiva

- Para cada patrón de entrenamiento:
  - Si la neurona da como salida -1, y debiera responder con 1, se debe incrementar el peso.
  - Si la neurona responde con +1,y debiera responder con -1, se debe decrementar el peso correspondiente.
  - En otro caso no modificar el valor del peso.

2.2. Algoritmo del perceptrón

1. **W=random()** *Asignar valores aleatorios a los pesos*
2. **neta = W*X** *Calcular la entrada neta*
3. **$y_{estimada}$ = f (neta)** *Aplicar la función umbral*
4. **$w_i = w_i + \Delta w_i$:** *Actualizar los pesos*

   Si $y_{estimada} < target$

   $\Delta w_i = + x_i \quad \forall i$ se debe incrementar el peso

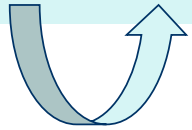
   Si_no Si $y_{estimada} > target$

   $\Delta w_i = - x_i \quad \forall i$ se debe decrementar el peso

   En otro caso

   $\Delta w_i = 0 \quad \forall i$ no se modifica el valor del peso
5. **Volver al paso 2**

Repetir hasta que se alcance un error aceptable o hasta que se llegue a un número determinado de ciclos.



2.3 Aprendizaje función AND

- Inicialización aleatoria de los pesos, por ejemplo:

  $w_1=1 \quad w_2=1 \quad w_0 = 0.5$

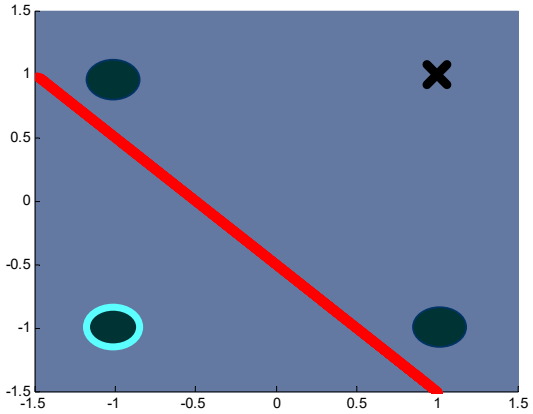
- Neta:

  $Neta = w_1 * x_1 + w_2 * x_2 + w_0$

  $Neta= (1 * -1) + (1*-1) + 0.5 = -1.5$

- Salida Calculada
  - $y= f(-1.5) = -1$
- Target
  - $t = -1$
- Clasificación correcta, no actualiza los pesos
- Volver al paso 2 con el siguiente patrón

x_1	x_2	target
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1



2.3 Aprendizaje función AND

- Procesamiento del segundo patrón:

  $w_1=1 \quad w_2=1 \quad w_0 = 0.5$

- Neta:
  - Neta = $w_1 * x_1 + w_2 * x_2 + w_0$
  - Neta= (1*-1)+ (1* 1) + 0.5 = 0.5
- Salida Calculada
  - y= f(0.5) = 1
- Target
  - t = -1
- Actualización de los pesos (decremento) $\Delta w_i = - x_i$

  $\Delta w_1 = 1 \Rightarrow w_1 = 1 + 1 = 2$

  $\Delta w_2 = -1 \Rightarrow w_2 = 1 - 1 = 0$

  $\Delta w_0 = -1 \Rightarrow w_0 = 0.5 - 1 = -0.5$

x_1	x_2	target
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

2.3 Aprendizaje función AND

- Procesamiento del tercer patrón:

  $w_1=2 \quad w_2=0 \quad w_0 = -0.5$

- Neta:
  - Neta = $w_1 * x_1 + w_2 * x_2 + \theta$
  - Neta= $2 * 1 + 0 * -1 - 0.5 = 1.5$
- Salida Calculada
  - $y= f(1.5) = 1$
- Target
  - $t = -1$
- Actualización de los pesos (decremento) $\Delta w_i = - x_i$

  $\Delta w_1= -1 \Rightarrow w_1 = 2 - 1 = 1$

  $\Delta w_2= 1 \Rightarrow w_2 = 0 + 1 = 1$

  $\Delta w_0= -1 \Rightarrow w_0 = -0.5 - 1 = -1.5$

x_1	x_2	target
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

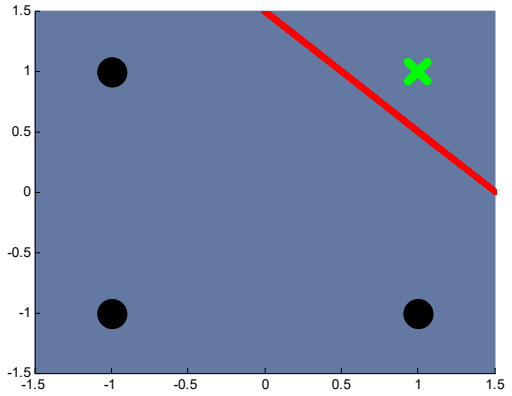
2.3 Aprendizaje función AND

- Procesamiento del cuarto patrón:

  $w_1=1 \quad w_2=1 \quad w_0 = -1.5$

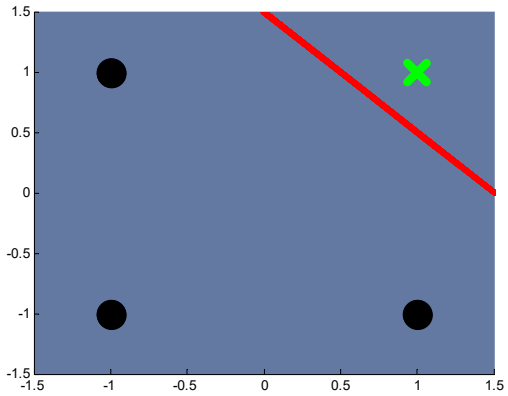
- Neta:
  - Neta= 1* 1 + 1 * 1 - 1.5 = 0.5
- Salida Calculada
  - y= f(0.5) = 1
- Target
  - t = 1
- No Actualiza

x_1	x_2	target
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1



2.3 Aprendizaje función AND

- Con los últimos pesos obtenidos hay que calcular la recta discriminante resultante y obtener el error de entrenamiento que se calcula sobre todos los patrones de entrada.

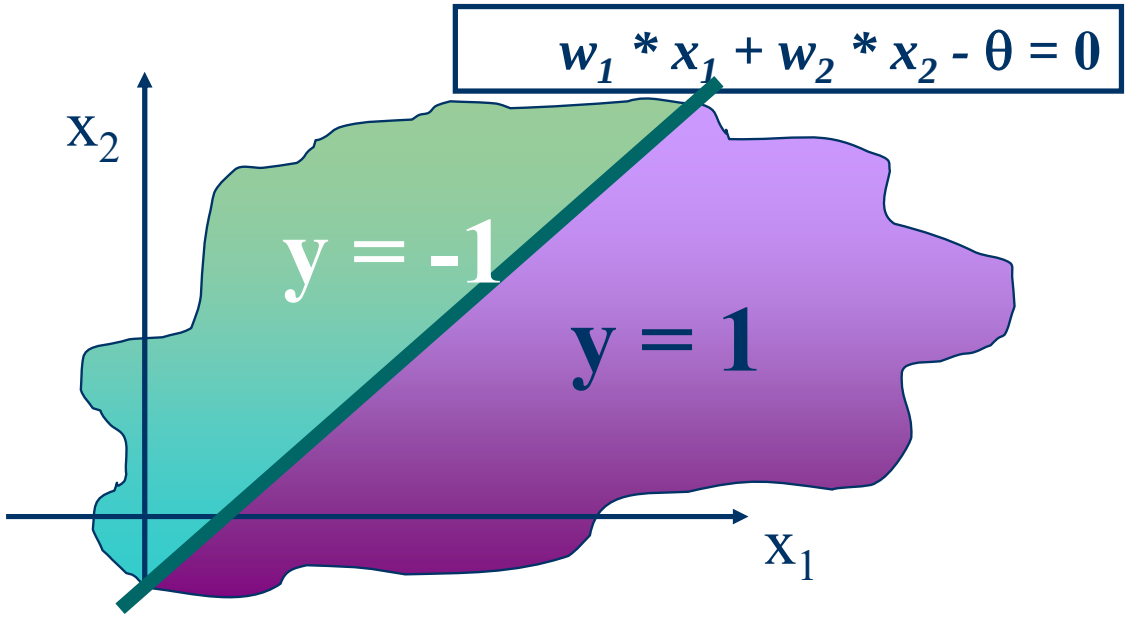


A	B	Target	Y Estimada
-1	-1	-1	-1
-1	1	-1	-1
1	-1	-1	-1
1	1	1	1

En este caso el error es 0 porque se han clasificado correctamente los 4 patrones.

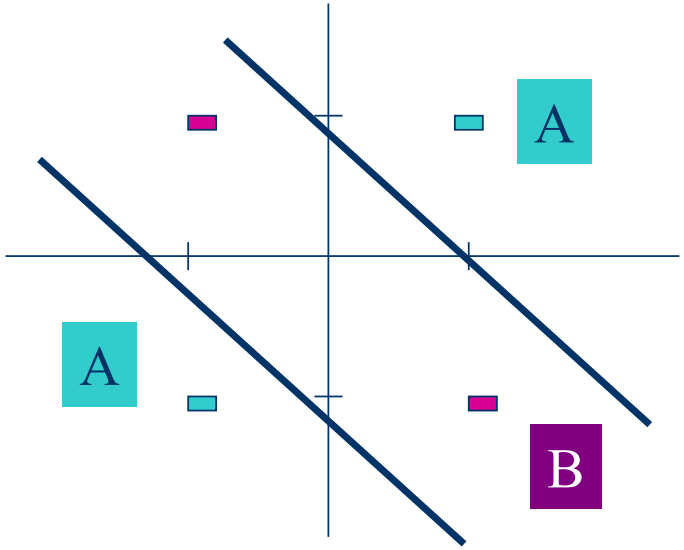
2.4 Teorema de convergencia

- Si las clases son linealmente separables, el algoritmo del perceptrón converge a una solución correcta en un número finito de pasos para cualquier elección inicial de pesos.



2.5 Limitaciones: función XOR

- No resuelven problemas tan sencillos como el Or-Exclusivo

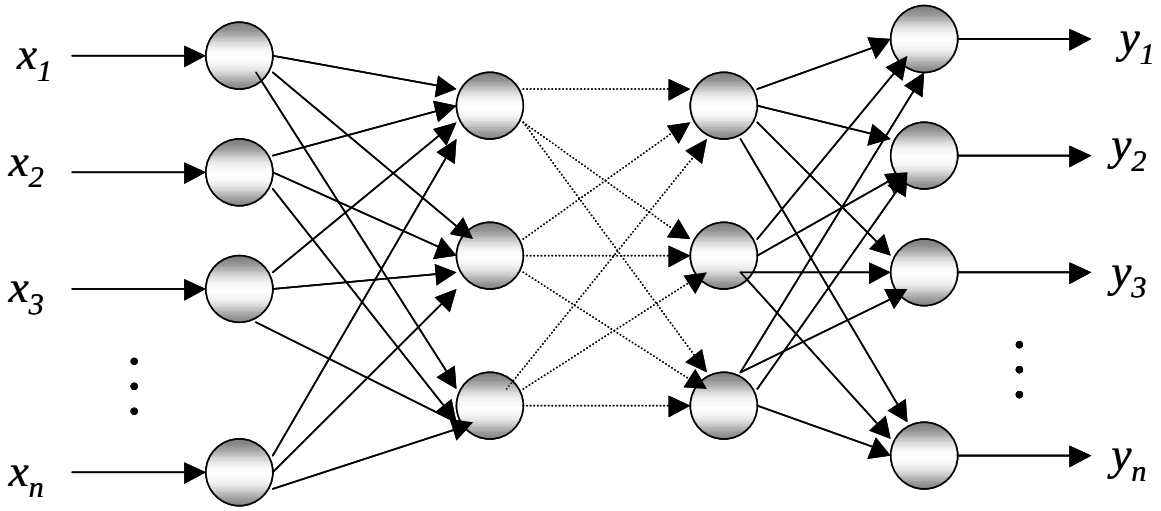


X_1	X_2	s = XOR (X)
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

- La mayoría de los problemas del mundo real no son linealmente separables

3. El Perceptrón Multicapa

- SOLUCIÓN => Añadir más capas



Capa de Entrada: **Reciben** patrones y los propagan a la siguiente capa

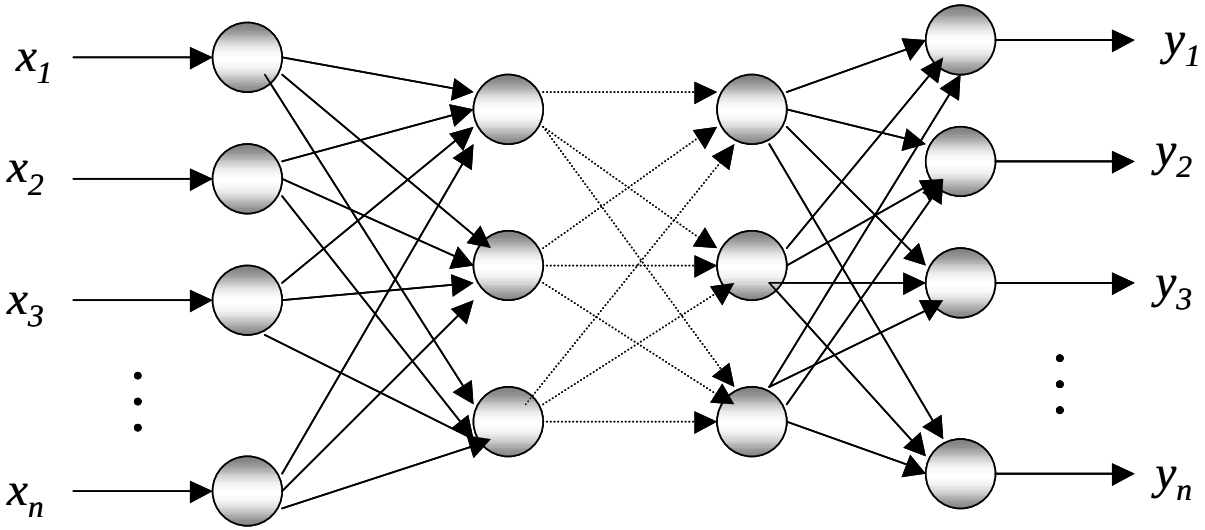
Capas Ocultas: Realizan **procesamiento no lineal**

Capa de Salida: Proporciona los **resultados**

3.1 Estructura del Perceptrón Multicapa

- Un Perceptrón Multicapa es capaz de aproximar cualquier función, siempre que las funciones de activación sean no lineales.

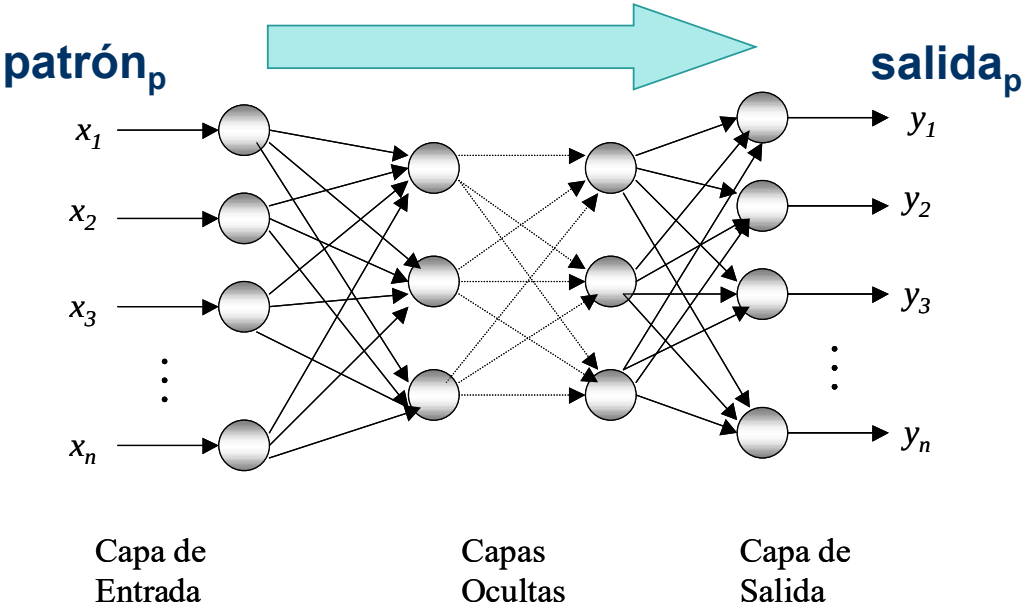
Las conexiones siempre están dirigidas hacia delante.



Las funciones de activación son No Lineales

3.2 Funcionamiento básico

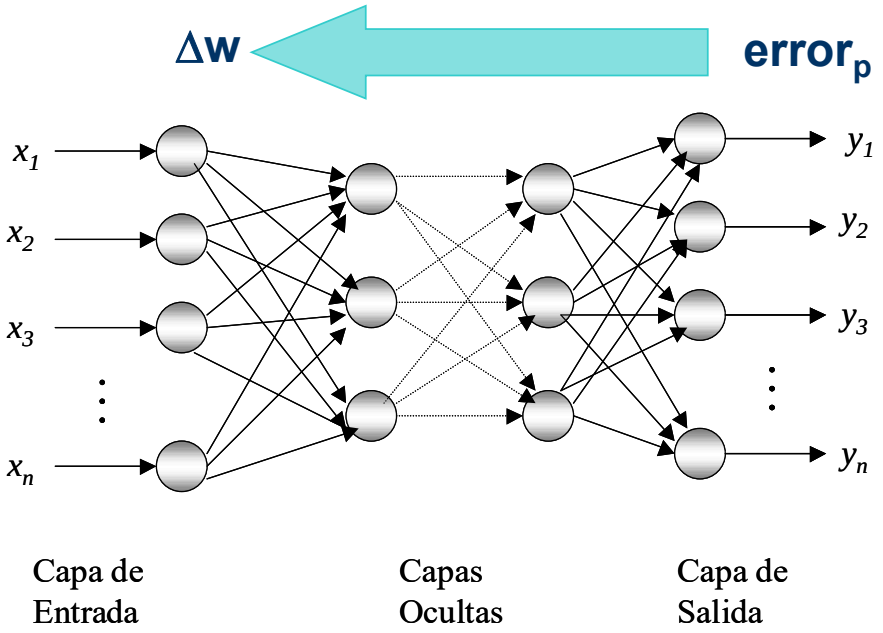
- 1ª Fase: **FORWARD**
  - Presentación de un patrón *p* de entrada
  - Procesamiento NO LINEAL en cada capa oculta
  - Transmisión a través de la red
  - Cálculo del Valor de Salida Estimado



3.2 Funcionamiento básico

2ª Fase: **BACWARD y OPTIMIZER**

- Cálculo del Error Cometido comparando el valor obtenido con el valor real (que es conocido).
- Propagación del error hacia atrás.
- Ajuste de los parámetros adaptativos.



3.2 Proceso de Aprendizaje

Inicialización aleatoria de los pesos.

REPETIR un número dado de epochs (ciclos de entrenamiento) o hasta obtener un error de validación aceptable:

ENTRENAMIENTO

Repetición con todos los patrones de entrenamiento.

1. FORWARD: Aplicar un vector de entrada x^p a la red y enviar hacia delante para calcular la salida predicha para ese patrón.

2. Evaluación de la función de pérdida (error), comparando con el valor deseado.

3. BACWARD: Retropropagación del error hacia atrás.

4. OPTIMIZER: cálculo de las derivadas de las funciones de activación en función de los pesos.

Actualización de los pesos.

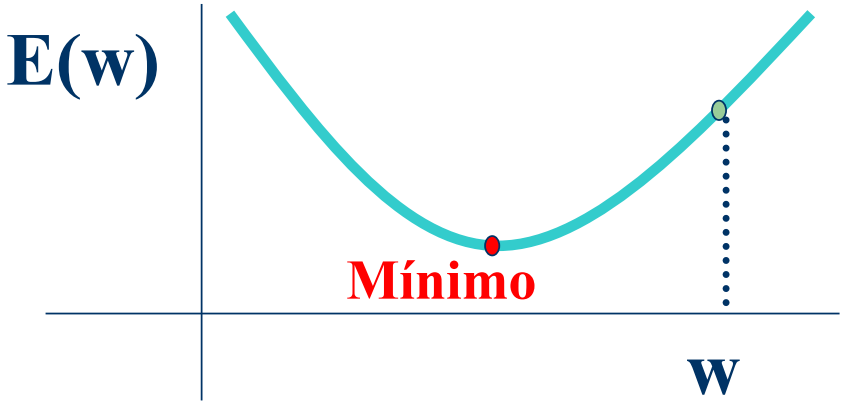
VALIDACIÓN

Cálculo del error con datos de Validación

Cálculo del Error total.

3.3 Algoritmo Backpropagation

- El objetivo es minimizar el error cometido entre la salida estimada y la salida deseada.
- Como no existe solución analítica conocida, es preciso utilizar un algoritmo iterativo.
- El algoritmo Backpropagation es un algoritmo iterativo que permite entrenar redes multicapa.

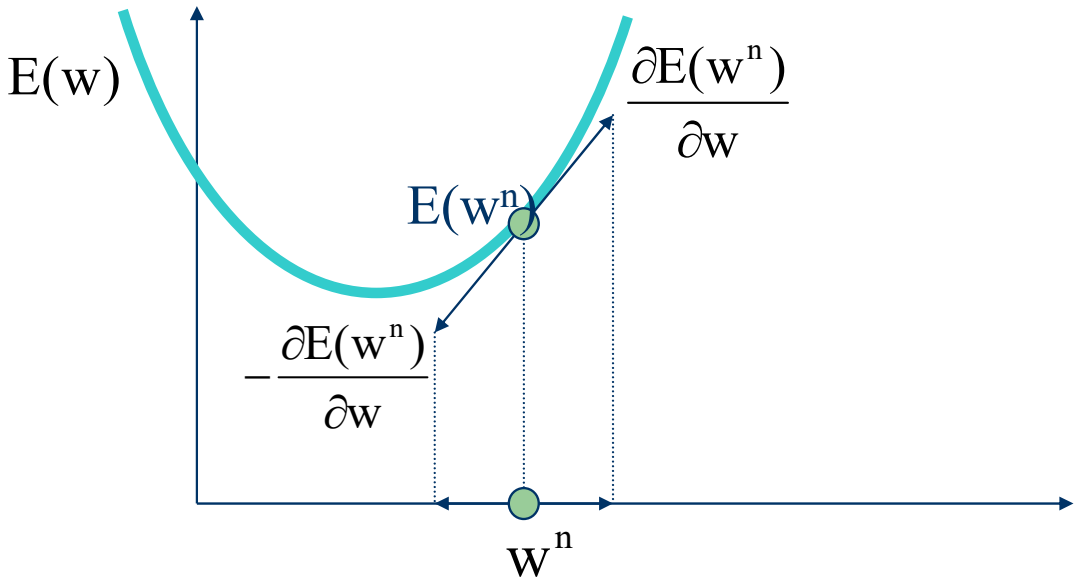


3.3 Algoritmo Backpropagation
Gradiente Descendente

- El gradiente es la pendiente de la tangente a E(w) en w= wi

Modificar los pesos proporcionalmente a la derivada de la función de error

$$\Delta w = -\alpha \frac{\partial E}{\partial w}$$



3. Algoritmo Backpropagation
Características fundamentales

Entrenar consiste en modificar los pesos de la red.

Necesita que la función de activación sea diferenciable (fácilmente).

Bacpropagation

Busca el mínimo de la función de error a partir de un conjunto de patrones de entrenamiento.

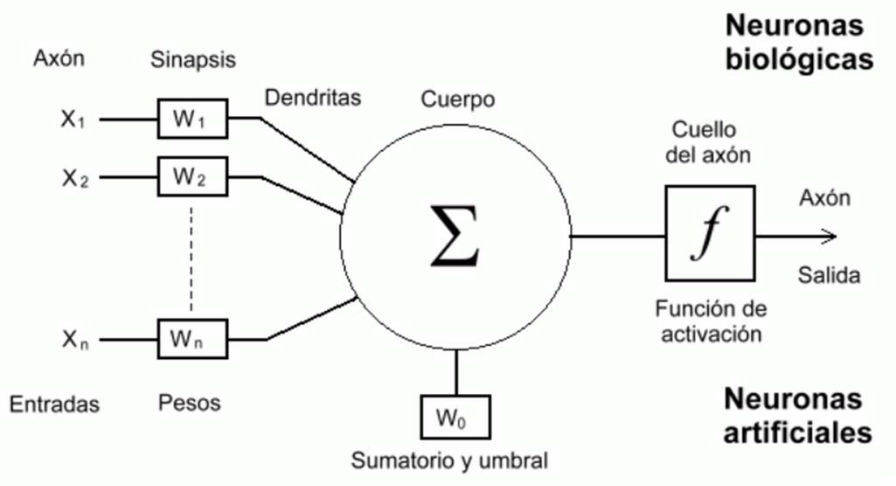
Los pesos se modifican hacia la dirección descendente de la función de error.

4. Funciones de Activación para las capas ocultas

La función de activación calcula la salida de la neurona en función de su entrada neta, por tanto realiza el modelado de la entrada a la salida (a partir de una suma ponderada).

- y = f(neta)
- yt = f(neta , yt-1)

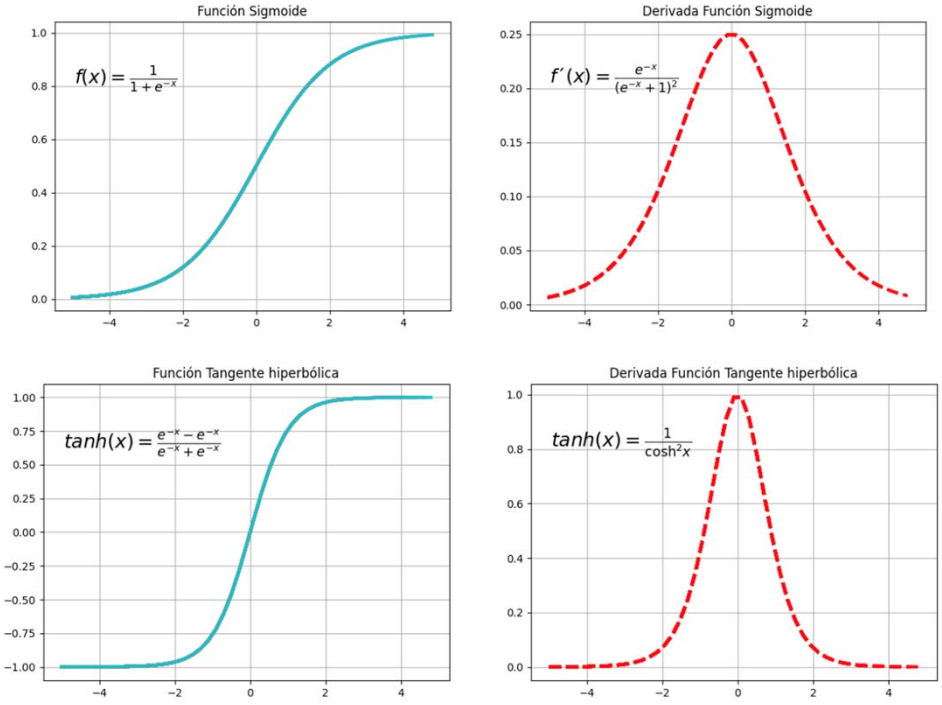
f { *Escalón*, *Lineal y mixta*, *Sigmoidal*, *ReLu*, *Gaussiana*, *Etc.* }



$$Y = f(w_0 + \sum_{i=1}^{n} w_i x_i)$$

4.1 Funciones de activación Sigmoide

- Sigmoide: toma cualquier rango de valores a la entrada y los mapea al rango de 0 a 1 a la salida.
- Tangente Hiperbólica: los valores de salida estarán en el rango de -1 a 1:



Sigmoide (logística)

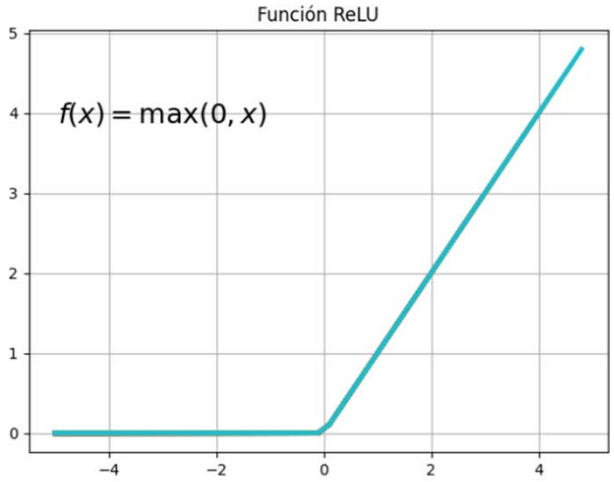
$$\phi(x) = \frac{1}{1 + e^{-ax}}$$

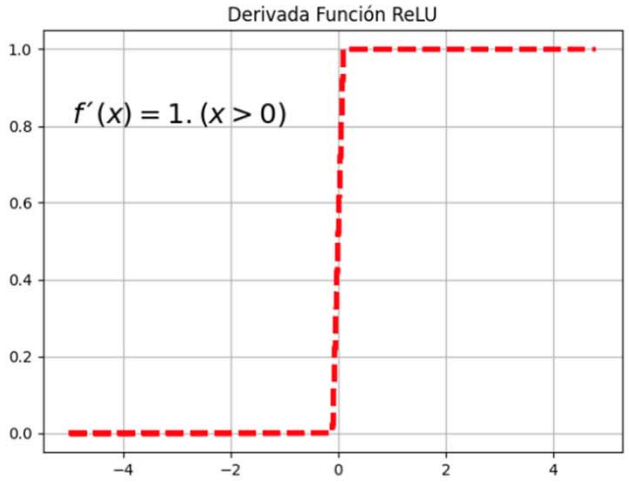
Tangente hiperbólica

$$\phi(x) = tanh(x)$$

4.2 ReLu *Rectified Linear Unit* (o unidad lineal rectificada).

- Elimina valores negativos: reemplaza todos los valores negativos recibidos en la entrada por ceros.
- Convergencia más rápida.
- Simplicidad en su implementación.





5. Función Softmax para la Capa de Salida

- Softmax o **función exponencial normalizada**: se utiliza en la clasificación multiclase para representar la probabilidad de cada una de las posibles clases de ser la inferida como resultado final del modelo de aprendizaje.
- Por tanto la distribución de probabilidad, que se encuentra entre 0 y 1, devuelve *K* diferentes valores y el máximo se tomará como la clase calculada por la red.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Capas Ocultas de la RNA

Capa de Salida SOFTMAX

0.72	GATO
0.12	PERRO
0.1	CONEJO
0.06	MONO

6. Funciones de pérdida o de error

- Indican el error cometido en la inferencia.
- **Problemas de regresión**:
  - Error cuadrático medio (MSE: Mean Squared Error)
  - Error absoluto medio (MAS: Mean Absolute Error)
- **Problemas de clasificación**:
  - Número de patrones mal clasificados (Accuracy)
  - Entropía cruzada:
    - Binary Cross-Entropy,
    - Categorical Cross-Entro

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

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i|$$

$$-(y \log(p) + (1 - y) \log(1 - p))$$

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

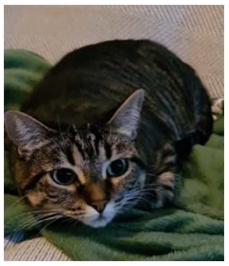
- M: número de clases
- log : log natural
- y :1 cuando es la clase deseada, 0 en otro caso
- p Probabilidad predicha

6.1 Entropía cruzada o “cross-entropy”

- Mide la diferencia entre dos distribuciones de probabilidad de un conjunto de eventos.
- Contiene un único mínimo, lo cual la hace ideal para su optimización usando el Gradiente Descendente

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \quad \text{=-(1*ln(0.72)+0*ln(0.12)+0*ln(0.1)+0*ln(0.06))=0.3285}$$

Cuanto más se asemejen la distribución generada y la distribución deseada, menor será la entropía cruzada de ambas.



Salida Calculada	Salida Real (target)
0.72	1
0.12	0
0.1	0
0.06	0

Puede tomar valores mayores que 1 en casos donde hay una discrepancia considerable entre las distribuciones de probabilidad real y estimada.

7. Evaluación del rendimiento del modelo entrenado
Matriz de Confusión

Tabla resumida que se utiliza para evaluar el rendimiento de un modelo de clasificación:

- **Positivo (P):** La observación es positiva (por ejemplo, es un gato).
- **Negativo (N):** La observación no es positiva (por ejemplo, no es un gato).
- **Verdadero Positivo (TP):** Resultado en el que el modelo predice correctamente la clase positiva.
- **Verdadero Negativo (TN):** Resultado donde el modelo predice correctamente la clase negativa.
- **Falso Positivo (FP):** También llamado error de tipo 1, donde el modelo predice incorrectamente la clase positiva cuando en realidad es negativa.
- **Falso Negativo (FN):** También llamado error de tipo 2, el modelo predice incorrectamente la clase negativa cuando en realidad es positiva.

7.1 Matriz de Confusión
Clasificación de 2 clases

		TARGETS	
		POSITIVO	**NEGATIVO**
PREDICCIONES	**POSITIVO**	CORRECTO (TP)	INCORRECTO (FP)
	NEGATIVO	INCORRECTO (FN)	CORRECTO (TN)

7.1 Matriz de Confusión
Ejemplo

- Supongamos un problema de clasificación binaria donde estás tratando de predecir si un correo electrónico es spam o no spam.
- Tras aplicar los datos de un conjunto de 100 datos de Test al modelo entrenado, se obtienen los siguientes resultados:

VP	31 correos electrónicos fueron correctamente clasificados como spam.
FP	4 correos electrónicos fueron incorrectamente clasificados como spam (falsos alarmas de spam).
VN	60 correos electrónicos fueron correctamente clasificados como no spam.
FN	5 correos electrónicos fueron incorrectamente clasificados como no spam (correos importantes que fueron considerados como no spam).

7.1 Matriz de Confusión
Clasificación de 2 clases

		TARGETS	
		SPAM	**NO SPAM**
PREDICCIONES	**SPAM**	**31** CORRECTO (TP)	**4** INCORRECTO (FP)
	NO SPAM	**5** INCORRECTO (FN)	**60** CORRECTO (TN)

7.2 Accuracy

- **Exactitud** (*Accuracy*): porcentaje de predicciones correctas frente al total.

$$Accuracy = \frac{\#\ of\ correct\ predictions}{total\ \#\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

		TARGETS	
		SPAM	NO SPAM
PREDICCIONES	SPAM	31	4
	NO SPAM	5	60

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{31 + 60}{31 + 60 + 4 + 5} = 0.91$$

91% de Exactitud en las Predicciones sobre datos nuevos

7.3 Precision

- **Precisión** (*Precision*): se refiere a lo cerca que está el resultado de una predicción del valor verdadero. Por tanto, es el cociente entre los casos positivos bien clasificados por el modelo y el total de predicciones positivas.

		TARGETS	
		SPAM	NO SPAM
PREDICCIONES	SPAM	31	4
	NO SPAM	5	60

$$Precision = \frac{TP}{TP + FP}$$

$$Precision = \frac{TP}{TP + FP} = \frac{31}{31 + 4} = 0{,}8857$$

La Precisión se enfoca en medir cuán preciso es un modelo en sus predicciones positivas.

7.4 Recall

- **Sensibilidad** (*Recall)*, tasa de verdaderos positivos. Es la proporción entre los casos positivos bien clasificados por el modelo, respecto al total de positivos, representa la capacidad de un modelo para discriminar los casos positivos, de los negativos.

		TARGETS	
		SPAM	NO SPAM
PREDICCIONES	SPAM	31	4
	NO SPAM	5	60

$$Recall = \frac{TP}{TP + FN}$$

$$Recall = \frac{TP}{TP + FN} = \frac{31}{31 + 5} = 0{,}8611$$

un Recall alto es deseable en situaciones donde la detección de casos positivos es crítica, como en aplicaciones médicas para detectar enfermedades o en la detección de fraudes en transacciones financieras.

7.5 Specifity

- **Especificidad** *(Specificity),* tasa de verdaderos negativos. Es la proporción entre los casos negativos bien clasificados por el modelo, respecto al total de negativos.

$$Specificity = \frac{TN}{TN + FP}$$

		TARGETS	
		SPAM	NO SPAM
PREDICCIONES	SPAM	31	4
	NO SPAM	5	60

$$Specifity = \frac{TN}{TN + FP} = \frac{60}{60 + 4} = 0{,}9375$$

Mide la proporción de correos electrónicos que realmente no son spam y que fueron correctamente clasificados como no spam.

7.6 F1 Score

Medida que combina tanto la precisión como la sensibilidad (recall) en una única métrica, y se calcula utilizando la fórmula:

$$F1\ score = \frac{2 * (precision * recall)}{precision + recall} = \frac{2TP}{2TP + FP + FN}$$

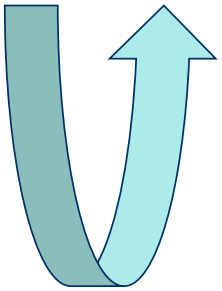
Cuanto más cercano a 1 mejor será la capacidad de generalización del modelo entrenado.

		TARGETS	
		SPAM	NO SPAM
PREDICCIONES	SPAM	31	4
	NO SPAM	5	60

$$F1score = \frac{2 * (Precision * Recall)}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} = \frac{2 * 31}{2 * 31 + 4 + 5} = 0{,}8732$$

8. Overfitting (Sobreajuste)

Datos de Entrenamiento	ENTRENAMIENTO
Datos de Validación	VALIDACIÓN
Datos de TEST	EVALUACIÓN FINAL

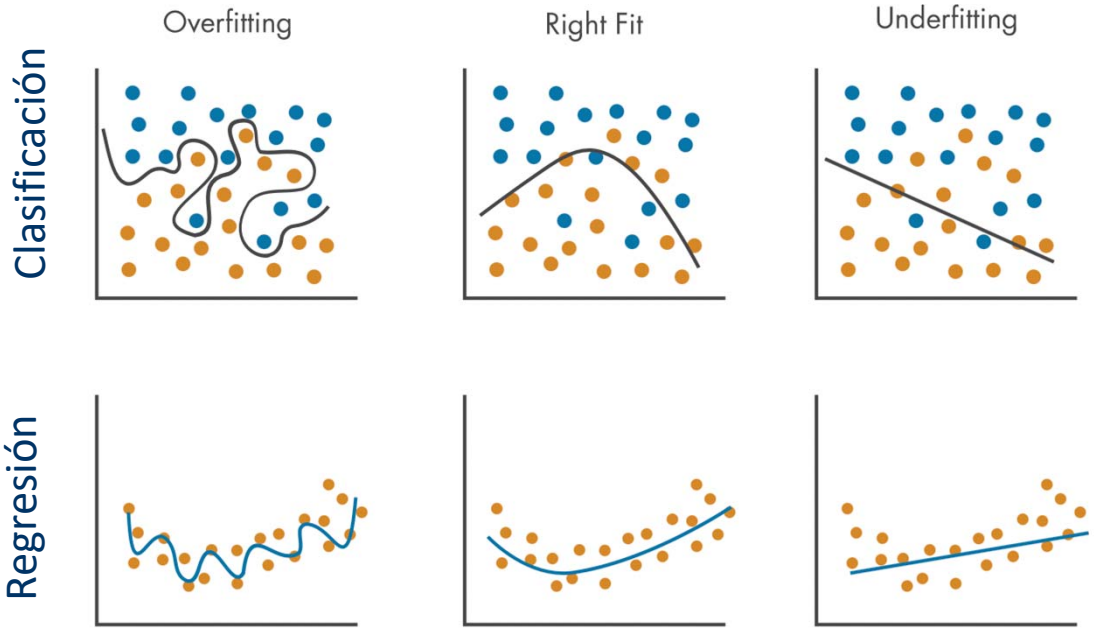


APRENDIZAJE

GENERALIZACIÓN

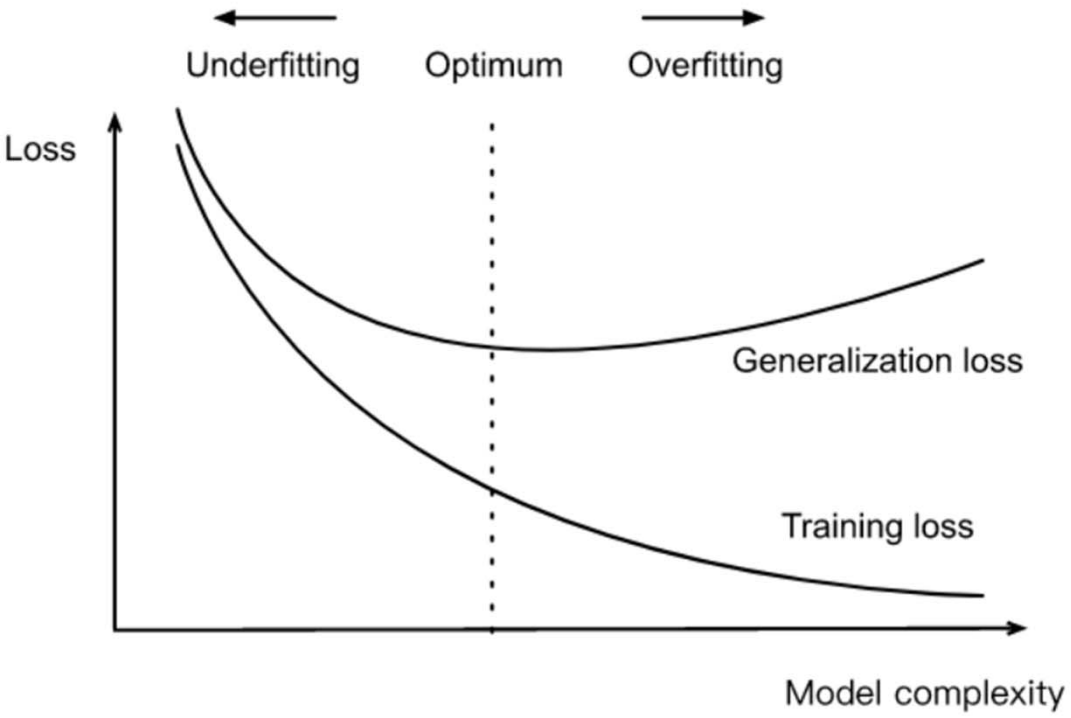
8. Overfitting (Sobreajuste)
¿Qué es?

Cuando el modelo entrenado, en lugar de enfocarse en los patrones generales que subyacen, presta una atención excesiva a todos los detalles, incluyendo aquellos que no tienen ningún impacto en la predicción.



- Afecta a cualquier modelo predictivo.
- Se pierde capacidad de generalización.

8. Overfitting (Sobreajuste)
Causas



- Complejidad
- Tiempo de entrenamiento
- Datasets

Referencias Bibliográficas

- Pattern Recognition and Machine Learning. **Bishop**, C. M. Springer, New York, NY, 2006.
- Neural Networks. A comprehensive foundation. S. **Haykin**, 1994
- Redes de Neuronas Artificiales. **Isasi** y Viñuela, 2003.
- Inteligencia Artificial: Un Enfoque Moderno. S. Russell y P. Norvig, 2005