

UNIVERSIDAD
NACIONAL
DE COLOMBIA

CLASIFICACIÓN Y RECONOCIMIENTO DE PATRONES

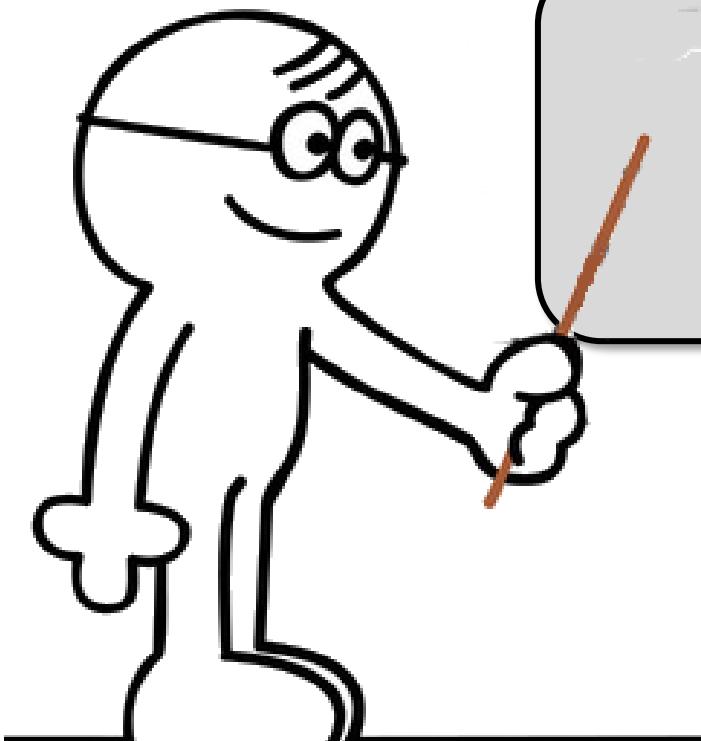
**Carlos Mera
John Willian Branch**

Departamento de Ciencias de la Computación y de la Decisión
Investigador del Grupo de I+D en Inteligencia Artificial – GIDIA

camerab@unal.edu.co

Contenido

1. Motivación
2. Métodos de Aprendizaje de Máquina:
 - a. Métodos Supervisados
 - b. Métodos No Supervisado
3. Regresión Lineal
4. Regresión Logística
5. KNN
6. Naïve Bayes
7. LDA y QDA
8. Máquinas de Vectores de Soporte
9. Árboles de Decisión
10. Redes Neuronales
11. Ensamblés de Clasificadores
12. Un ejemplo práctico: detección de defectos en dientes

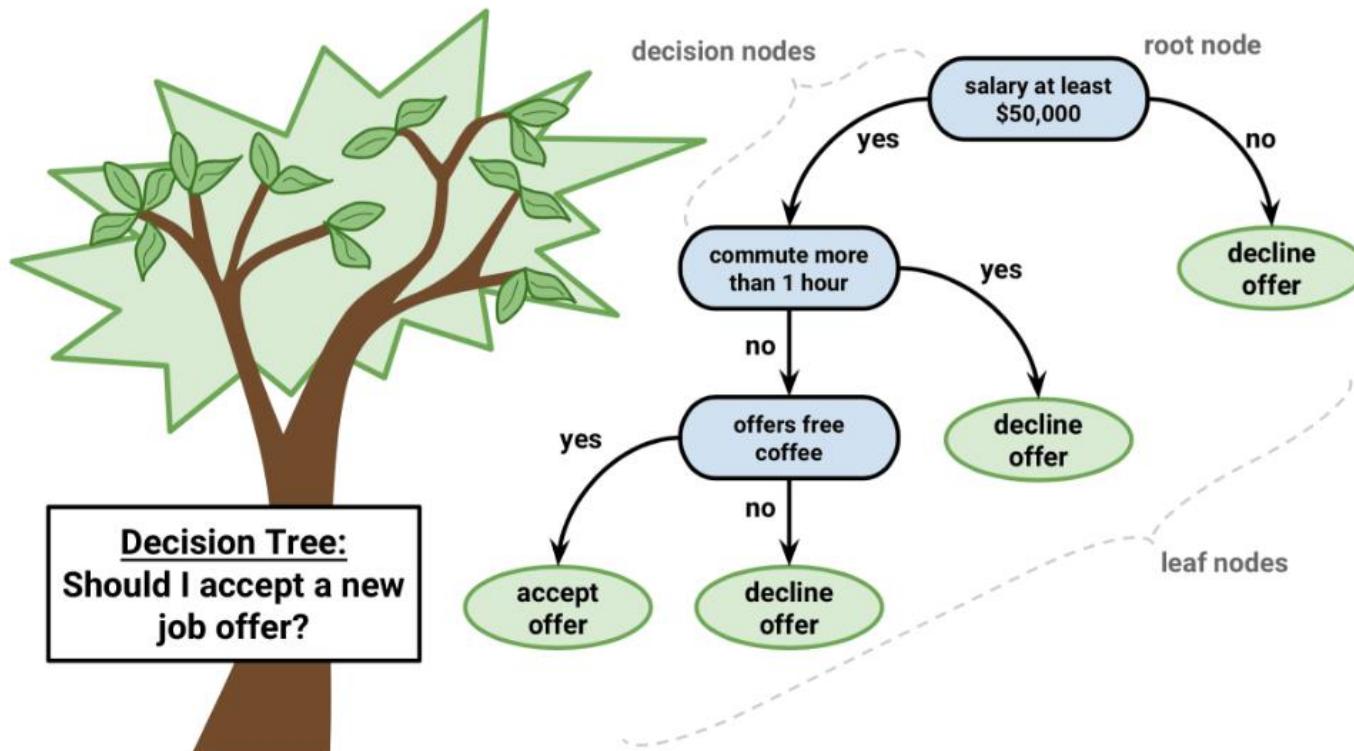


ÁRBOLES DE DECISIÓN

Árboles de Decisión

DEFINICIÓN:

Es una técnica de aprendizaje que construye un árbol para clasificar un grupo de instancias tomando decisiones con base en los valores de sus atributos.

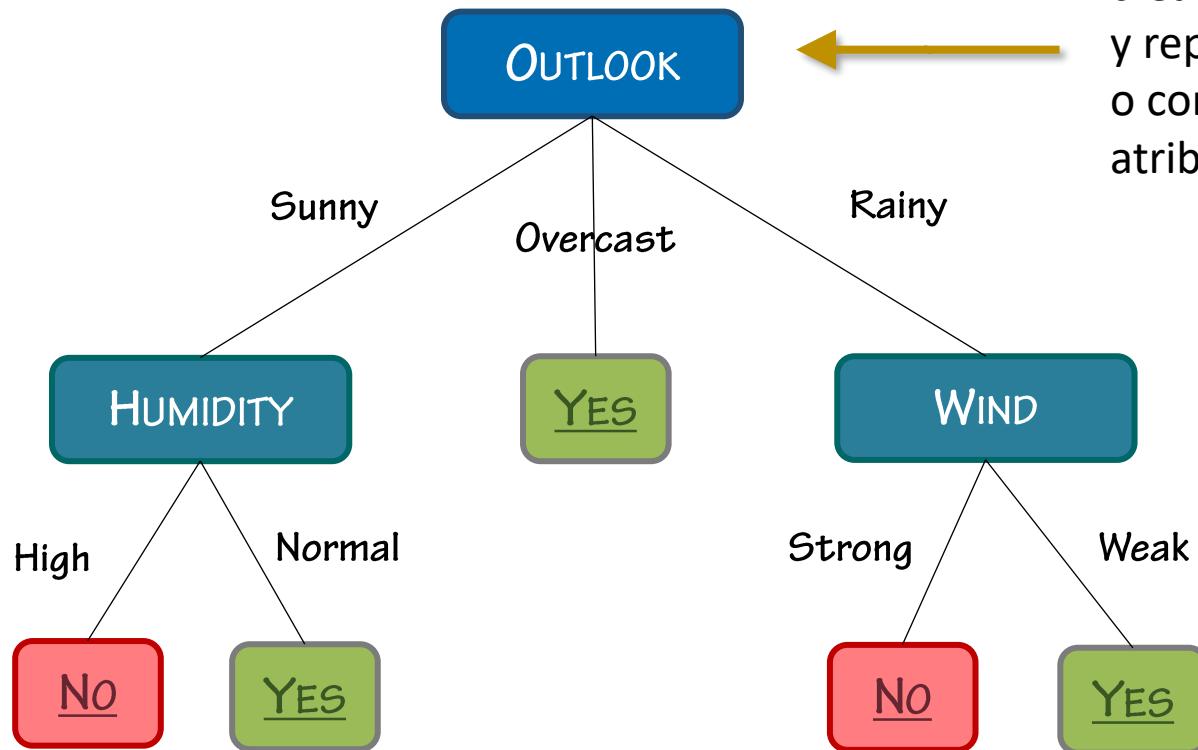


<https://bookdown.org/content/2031/arboles-de-decision-parte-i.html>

Árboles de Decisión

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Las hojas
determinan la
predicción



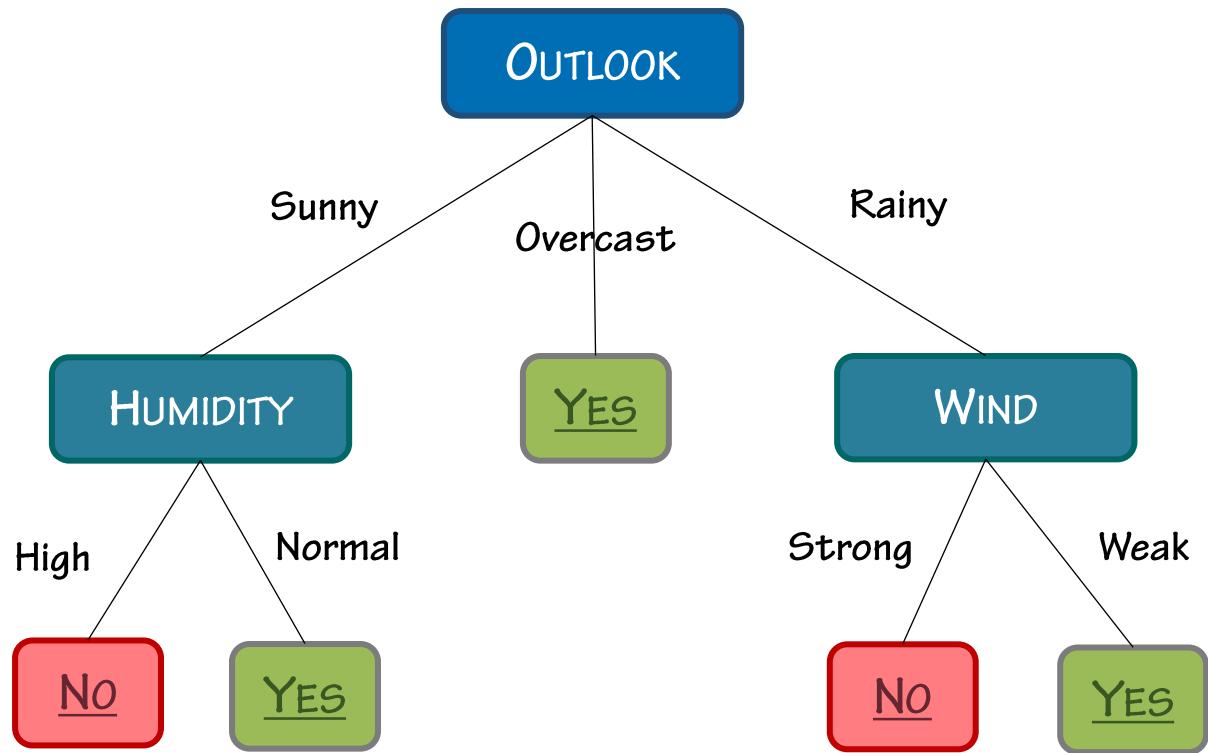
Los nodos intermedios
crean las ramas del árbol
y representan decisiones
o condiciones sobre un
atributo.

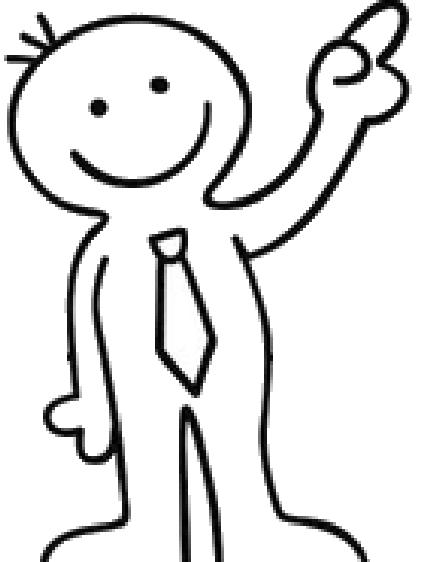
Árboles de Decisión

¿CÓMO FUNCIONA?:

Para clasificar un ejemplo, se recorre el árbol desde su raíz hasta llegar a una hoja, evaluando la condición que se encuentra en cada nodo y siguiendo la rama de acuerdo con el resultado de la condición. Cuando se alcanza una hoja se retorna la clase que esta define.

Con base en este árbol, ¿se decide jugar o no al tenis si el viento está fuerte, la humedad normal y el día soleado?





¿CÓMO SE CONSTRUYE EL ÁRBOL?

Árboles de Decisión



¿CÓMO SE CONSTRUYE EL ÁRBOL?:

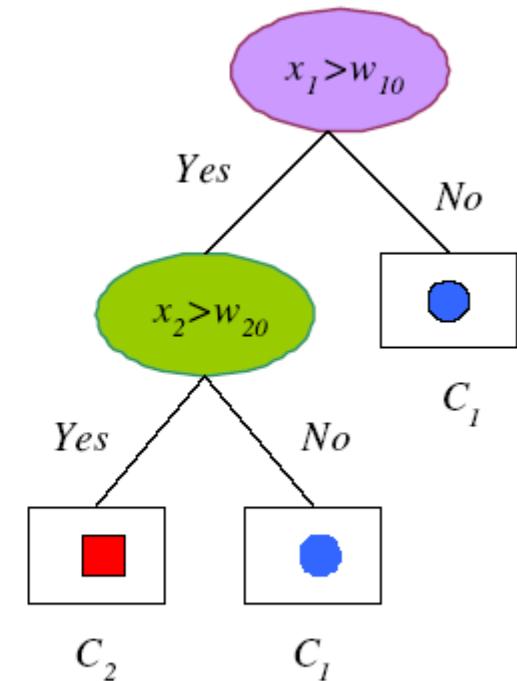
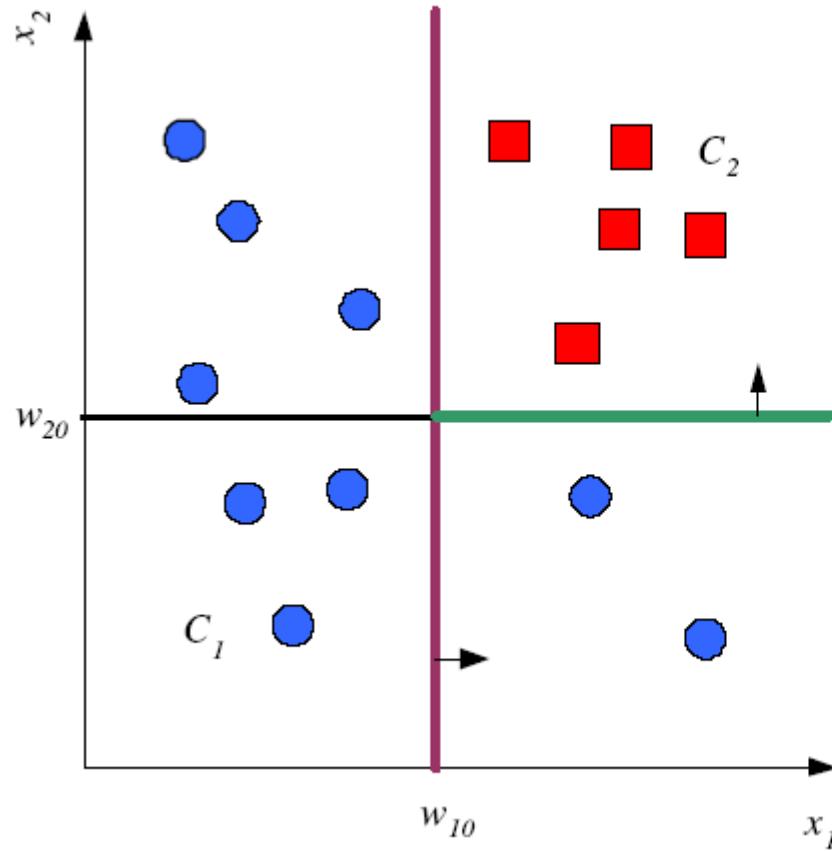
1. Inicialmente todos las muestras en el conjunto de entrenamiento se usan para definir el atributo inicial de ramificación que va en la raíz del árbol.
2. Una vez se selecciona el atributo de ramificación, las muestras se dividen en función del atributo seleccionado y se pasan al siguiente nodo para repetir el proceso.
3. La selección del atributo de cada nodo se hace en función de una heurística que determina cuál es el mejor atributo o cuál es el atributo que proporciona más información para hacer la división.
4. El proceso se repite hasta que todos los ejemplos en un nodo sean de la misma clase o no queden atributos para hacer la división, en cuyo caso se crea una hoja con la clase más frecuente.

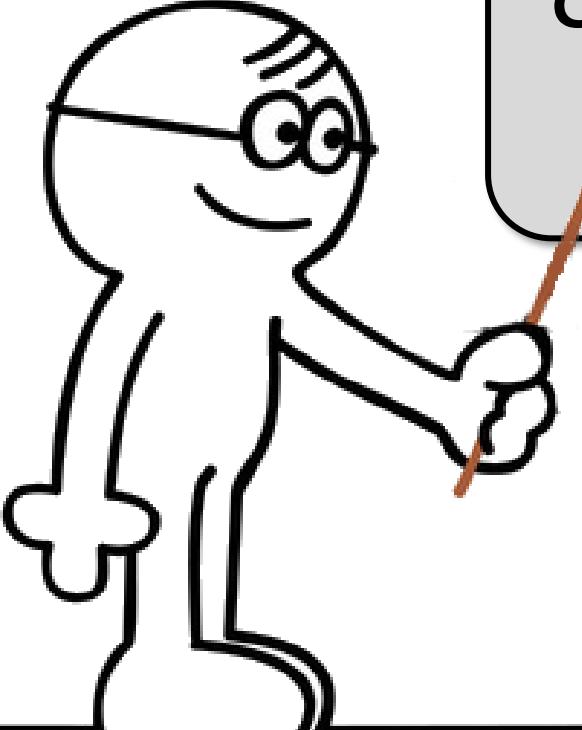
Es un algoritmo VORAZ!!

Árboles de Decisión

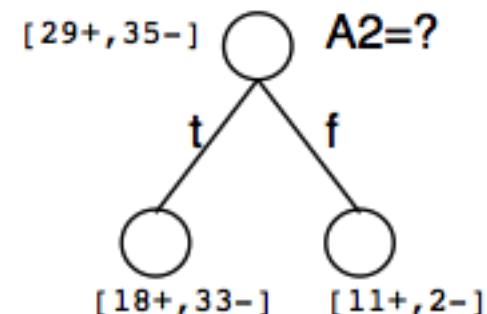
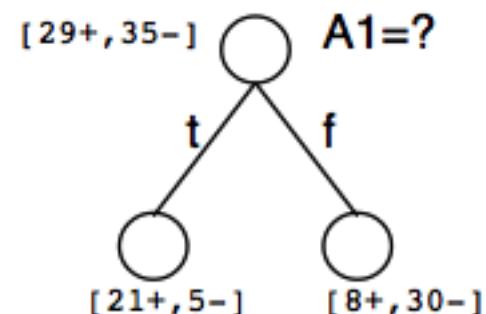
LAS FRONTERAS DE DECISIÓN DE UN ÁRBOL:

Las fronteras de decisión se construyen con base en particiones rectangulares





¿CÓMO DECIDIR CUÁL ES EL MEJOR ATRIBUTO?



Árboles de Decisión

CRITERIO DE PARTICIÓN:

La idea detrás de ramificar el árbol es conseguir nodos más y más **puros** (u homogéneos) tras las divisiones recursivas. Así, para un nodo m con N_m instancias, N_m^i pertenecen a la clase C_i se tiene que:

$$\hat{P}(C_i|m) \equiv p_m^i = \frac{N_m^i}{N_m}$$

Con base en esta probabilidad y para un problema de dos clase, se dice que el nodo m **es puro** si p_m^i es 0 o 1. Así, en la construcción del árbol, el atributo usado para hacer la partición siempre es el que genera el nodo más puro.

Existen diferentes medidas de pureza o impureza:

- La entropía
- El índice de Gini
- Un test Chi Cuadrado
- Reducción de varianza

Árboles de Decisión



CRITERIO DE PARTICIÓN - GANANCIA DE INFORMACIÓN (ALGORITMOS ID3, C4.5)

La **ganancia de información** mide cuán bien una característica separa los ejemplos de entrenamiento de acuerdo a los etiquetas de salida.

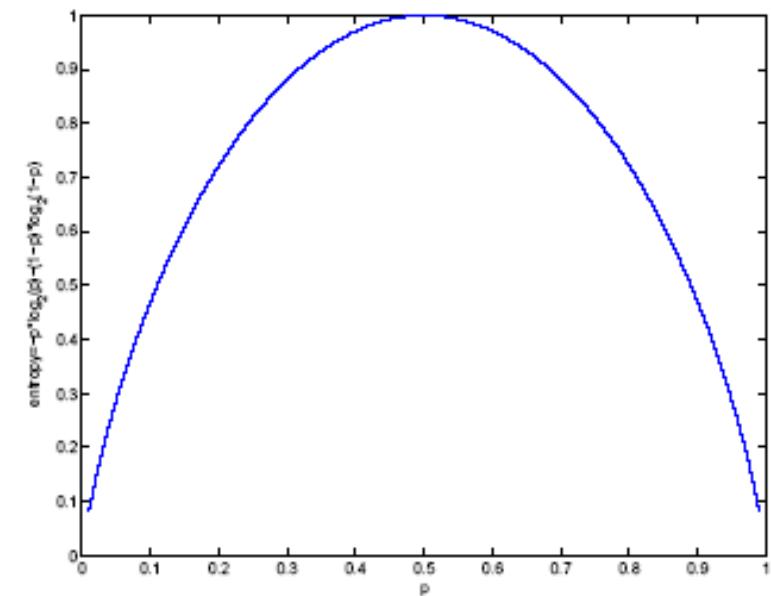
La **entropía** es una medida de impureza que puede usarse para medir la ganancia de información al particionar un conjunto de datos usando un atributo.

En un problema de clasificación binario se tiene

- El nodo S tiene m instancias de entrenamiento
- p^+ es la proporción de instancias positivas en S
- p^- es la proporción de instancias negativas en S

La entropía del nodo S se calcula cómo:

$$\text{Entropia}(S) = -p^+ \log_2(p^+) - p^- \log_2(p^-)$$



Árboles de Decisión



CRITERIO DE PARTICIÓN - GANANCIA DE INFORMACIÓN (ALGORITMOS ID3, C4.5):

En un problema de clasificación multiclase en el que p_i es la probabilidad de ocurrencia de la clase i en el nodo S , la entropía se calcula como:

$$H(x) = - \sum_{i=1}^c p_i \log_2 p_i = \sum_{i=1}^c p_i \log_2(1/p_i)$$

Cuanto mayor es la entropía, mayor es la incertidumbre. Así, la ganancia de información se define como la reducción esperada de la entropía causada por la partición de los datos en un atributo. Esto es:

$$Gain(S, A) = Entropia(S) - \sum_{v=1}^{\nu} \frac{|S_v|}{|S|} Entropia(S_v)$$

Donde, S_v es el subconjunto de instancias que tienen el valor v , en el atributo A .

Árboles de Decisión



CRITERIO DE PARTICIÓN - GANANCIA DE INFORMACIÓN (ALGORITMOS ID3, C4.5):

Veamos un ejemplo, para determinar el atributo en la raíz del árbol tenemos:

- Valores para el atributos wind = {weak, strong}
- $S = [9+, 5-]$
- $S_{Weak} = [6+, 2-]$
- $S_{Strong} = [3+, 3-]$

Ganancia de información de wind:

$$\begin{aligned}Gain(S, Wind) &= Entropy(S) - \frac{8}{14} Entropy(S_{Weak}) - \frac{6}{14} Entropy(S_{Strong}) \\&= 0,94 - \frac{8}{14} \times 0,811 - \frac{6}{14} \times 1,0 \\&= 0,048\end{aligned}$$

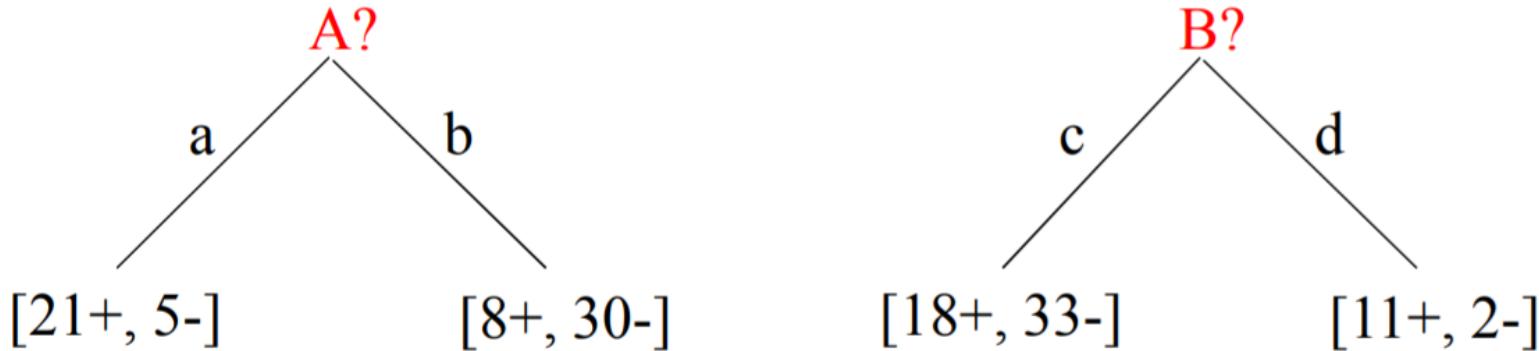
Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Árboles de Decisión

💡 CRITERIO DE PARTICIÓN - GANANCIA DE INFORMACIÓN (ALGORITMOS ID3, C4.5):

Otro ejemplo: ¿Cuál es mejor, el atributo A o el atributo B?

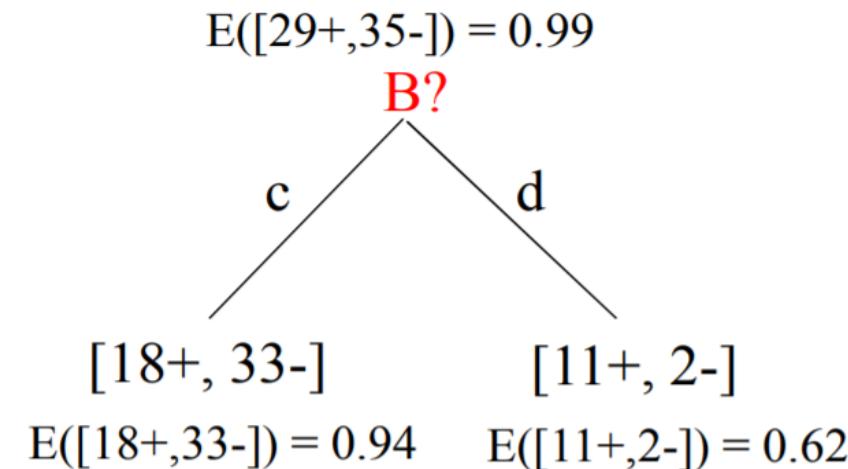
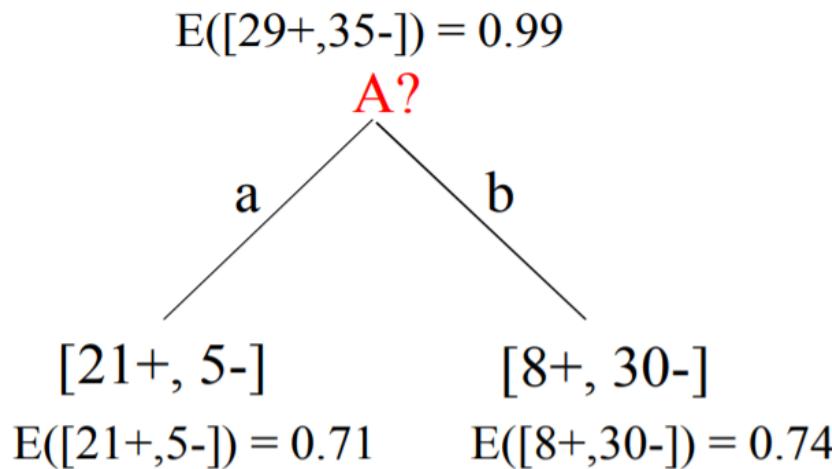
- S: [29+,35-] Attributes: **A** and **B**
- possible values for A: a,b possible values for B: c,d
- Entropy([29+,35-]) = $-29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$



Árboles de Decisión

CRITERIO DE PARTICIÓN - GANANCIA DE INFORMACIÓN (ALGORITMOS ID3, C4.5):

Otro ejemplo: ¿Cuál es mejor, el atributo A o el atributo B?



$$\begin{aligned} \text{Gain}(S,A) &= \text{Entropy}(S) \\ &\quad -26/64 * \text{Entropy}([21+,5-]) \\ &\quad -38/64 * \text{Entropy}([8+,30-]) \\ &= \mathbf{0.27} \end{aligned}$$

$$\begin{aligned} \text{Gain}(S,B) &= \text{Entropy}(S) \\ &\quad -51/64 * \text{Entropy}([18+,33-]) \\ &\quad -13/64 * \text{Entropy}([11+,2-]) \\ &= \mathbf{0.12} \end{aligned}$$

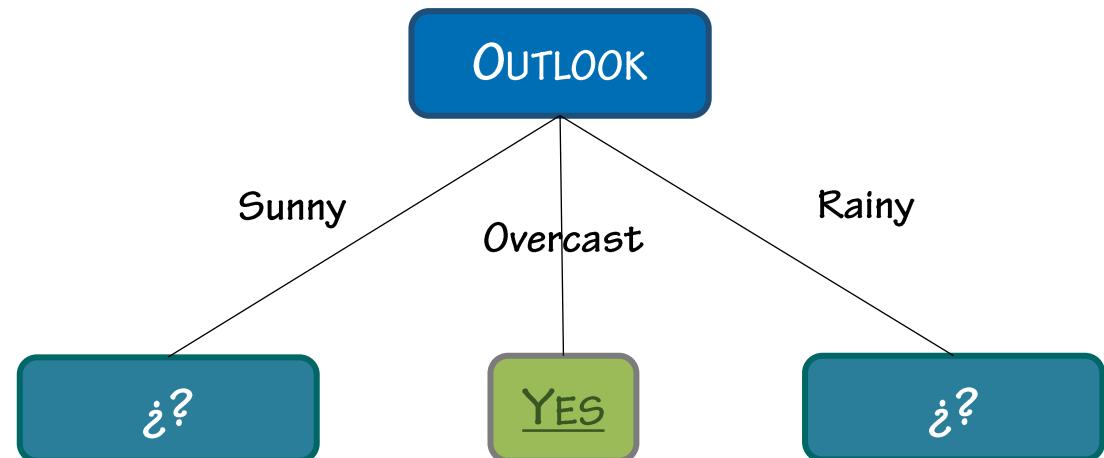
Árboles de Decisión

CRITERIO DE PARTICIÓN - GANANCIA DE INFORMACIÓN (ALGORITMOS ID3, C4.5):

Considerando todos los atributos tenemos: ¿Qué atributo debe probarse en la raíz?

- $Gain(S, Outlook) = 0.246$
- $Gain(S, Humidity) = 0.151$
- $Gain(S, Wind) = 0.084$
- $Gain(S, Temperature) = 0.029$

De acuerdo con la métrica de ganancia de información
Outlook es el atributo que proporciona la mejor
predicción. Así que se crea la primera partición
con base en los tres valores de este atributo:



Árboles de Decisión

CRITERIO DE PARTICIÓN - ÍNDICE DE GINI (ALGORITMO CART):

El índice de Gini también es otra medida de impureza, la cual se calcula como:

$$Gini(S) = 1 - \sum_{i=1}^c p_i^2$$

Como ejemplo, analice las siguientes mediciones para un problema de dos clases:

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Para construir el árbol, se elige el atributo con la menor impureza:

$$I(S, A) = \sum_{i=1}^v \frac{|S_v|}{|S|} Gini(S_v)$$

Árboles de Decisión

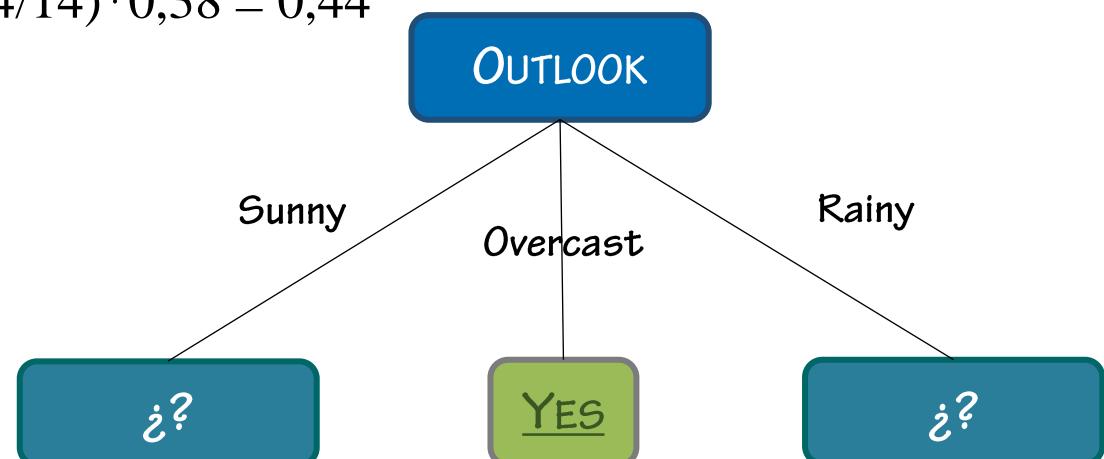


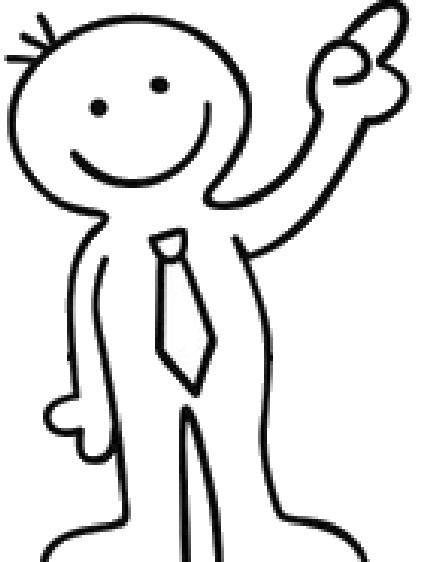
CRITERIO DE PARTICIÓN - ÍNDICE DE GINI (ALGORITMO CART):

Considerando todos los atributos tenemos: ¿Qué atributo debe probarse en la raíz?

- $Gini(S, Outlook) = (5/14)*0,48 + (4/14)*0+ (5/14)*0,48 = 0,34$
- $Gini(S, Humidity) = (7/14)*0,49 + (7/14)*0,25 = 0,37$
- $Gini(S, Wind) = (8/14)*0,38 + (6/14)*0,5 = 0,43$
- $Gini(S, Temperature) = (4/14)*0,5 + (6/14)*0,44 + (4/14)*0,38 = 0,44$

De acuerdo con la métrica, el atributo que menos información pierde es Outlook. Así que se crea la primera partición con base en los tres valores de este atributo.





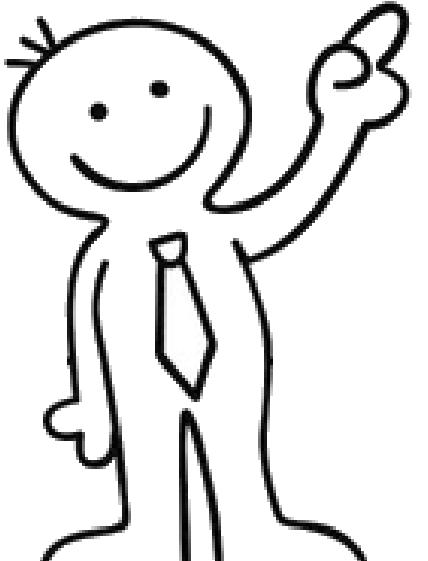
¿Y QUÉ PASA CUANDO LOS
ATRIBUTOS SON CONTINUOS?

Árboles de Decisión

CON ATRIBUTOS CONTINUOS:

Hay diferentes formas de manejarlo:

- **Discretización:** para formar un atributo categórico ordinal
 - Estática: desde el inicio se definen el número de categorías para el atributo
 - Dinámica: se definen rangos para crear grupos de tamaños iguales, se hace uso de los percentiles o incluso se puede aplicar un algoritmo de *clustering* para hacer la discretización.
- **Decisión binaria del tipo $(A < v)$ o $(A \geq v)$:**
En este caso se deben considerar todas las divisiones posibles buscando el mejor punto de corte. Sin embargo, este tipo de soluciones requieren más tiempo por los cálculos a realizar.



¿Y CUÁL ES EL CRITERIO DE
PARADA?

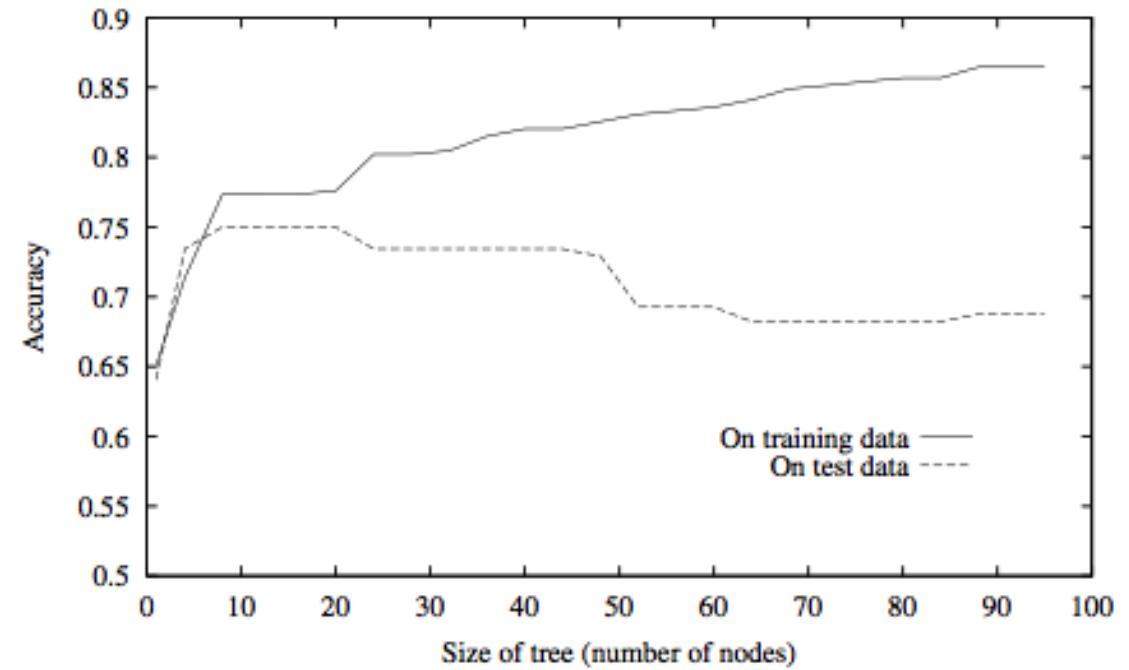
Árboles de Decisión

CRITERIOS DE PARADA BÁSICOS:

- Pare la expansión de una rama cuando todas las instancias en el nodo sean de la misma clase
- Pare la expansión cuando no hay más atributos para evaluar
- Pare la expansión cuando todas las instancias tienen el mismo valor para e atributo en nodo

PROBLEMA DE SOBREAJUSTE

Los árboles de decisión tienden a ajustarse demasiado al conjunto de entrenamiento utilizado para construir del árbol, cuando este se expande por completo!



Árboles de Decisión

SOLUCIONES AL SOBREAJUSTE:

- Pre-poda (frenar la expansión del árbol):
 - Se decide no expandir el árbol cuando solo queden x instancias en el nodo
 - Se decide no expandir el árbol cuando se ha llegado a una profundidad máxima definida
- Post-poda (expanda el árbol y luego recórtelo):
 - Recorte los nodos del árbol de abajo hacia arriba
 - Si el error de generalización mejora después del recorte, reemplace el subárbol por un nodo hoja.
La etiqueta de clase del nodo hoja se determina a partir de la clase mayoritaria en el nodo

Árboles de Decisión



VENTAJAS:

- Es un modelo de clasificación simple y sus resultados son completamente interpretables.
- Es un modelo de clasificación robusto al ruido
- Es capaz de aprender de atributos continuos y categóricos



DESVENTAJAS:

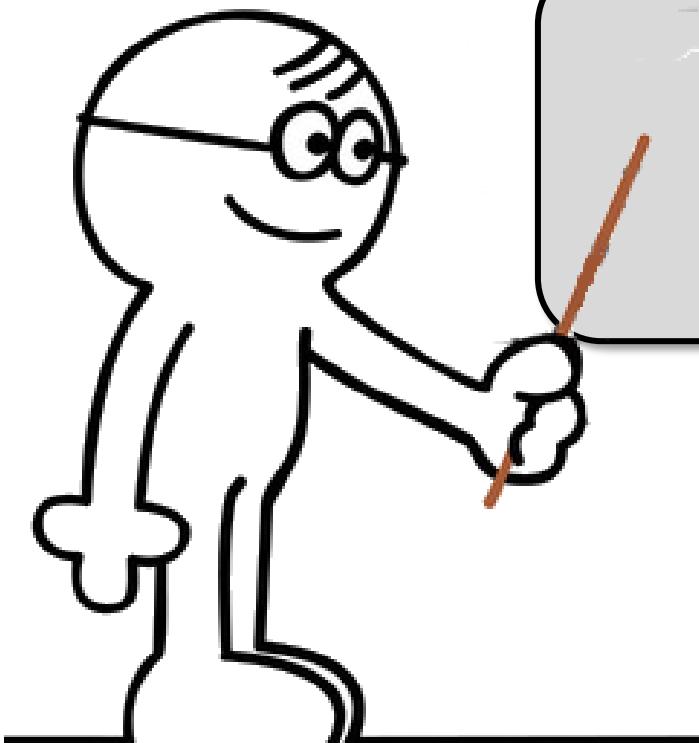
- Fronteras de decisión rectangulares que pueden no ser buenas para algunos conjuntos de datos.
- Seleccionar buenos parámetros de algoritmo de aprendizaje (por ejemplo, grado de poda) no es trivial
- Más recientemente, los bosques (conjuntos de árboles de decisión) han ganado cierta popularidad.





VAMOS A CODIFICAR!

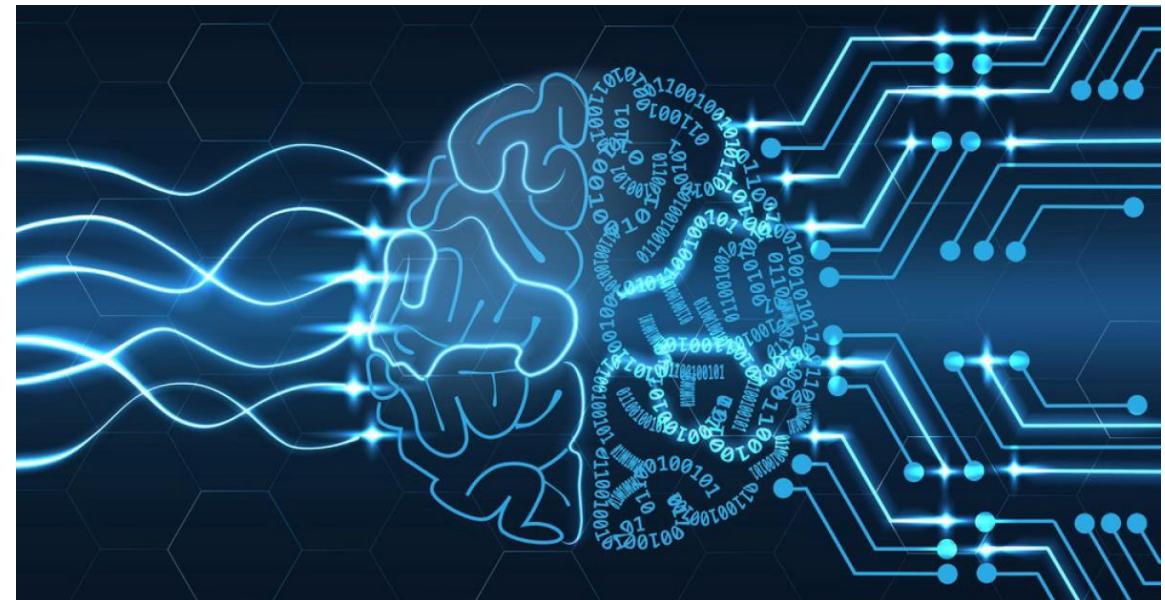
REDES NEURONALES



Redes Neuronales

MOTIVACIÓN:

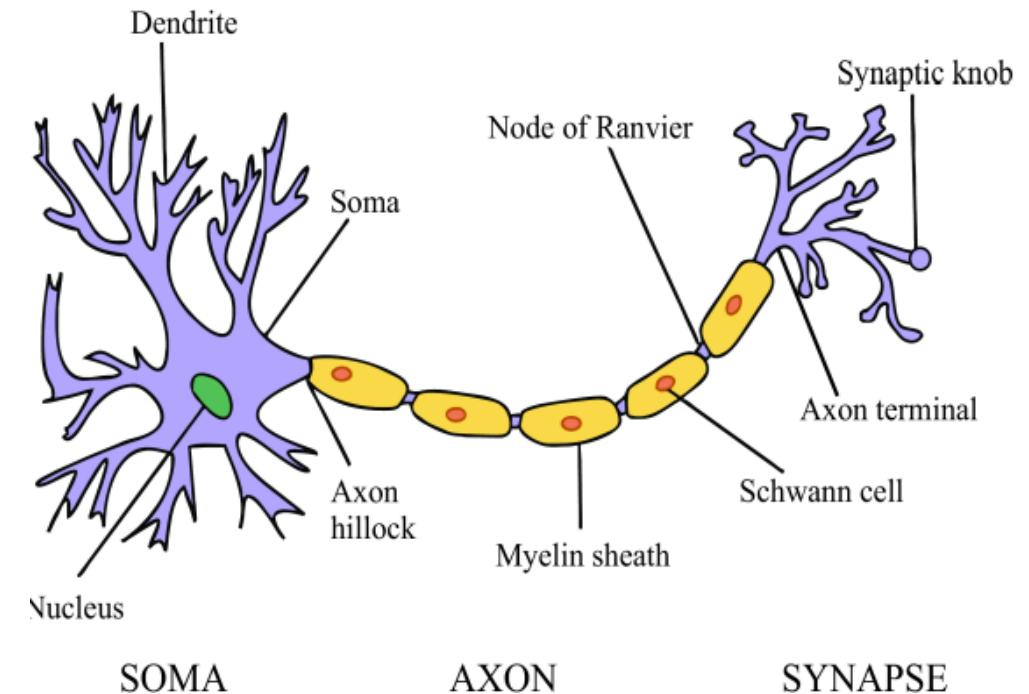
- El cerebro es una máquina altamente compleja, no lineal y trabaja en forma paralela para procesar la información. Adicionalmente, es capaz de organizar su estructura para desarrollar ciertos cómputos (reconocimiento de patrones, percepción y control motor) de forma muy rápida.
- La habilidad del cerebro para construir reglas (aprender) es a través de la experiencia.
- Una red neuronal modela la forma en la que el cerebro se desempeña en una tarea particular.
- Esa red utiliza una **interconexión masiva** de células de cómputo simple, llamadas neuronas o unidades de procesamiento.



Redes Neuronales

DEFINICIONES:

- La unidad básica de procesamiento es la **neurona** y la clave para el procesamiento de la información son las conexiones entre las neuronas, a lo que se llama **sinapsis**.
- En una red neuronal la fortaleza de las conexiones inter-neuronas, que son los pesos sinápticos, se usan para almacenar el conocimiento adquirido.
- El aprendizaje entonces consiste en modificar los pesos sinápticos de la red con el fin de obtener un desempeño deseado en una tarea.

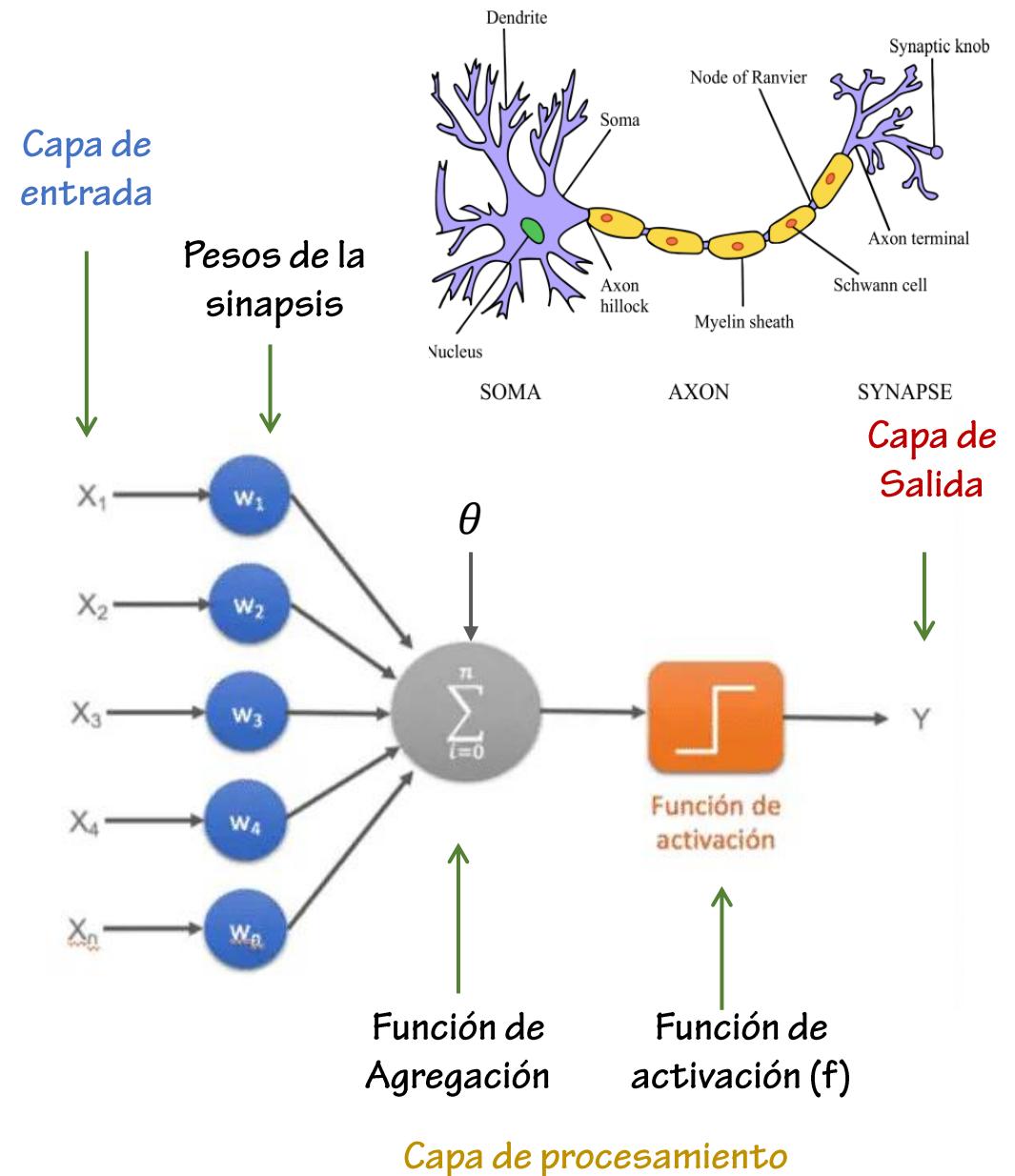


Redes Neuronales

PERCEPTRÓN SIMPLE:

- En el caso de una red neuronal artificial, la **neurona** tiene una estructura similar a una neurona natural, la cual se conoce como el **perceptrón simple**.

El funcionamiento del perceptrón es muy sencillo, simplemente toma los valores en la capa de entrada y hace una suma ponderada de acuerdo con los pesos de cada entrada. Luego, el resultado lo introduce en una función de activación que genera el resultado final.



Redes Neuronales

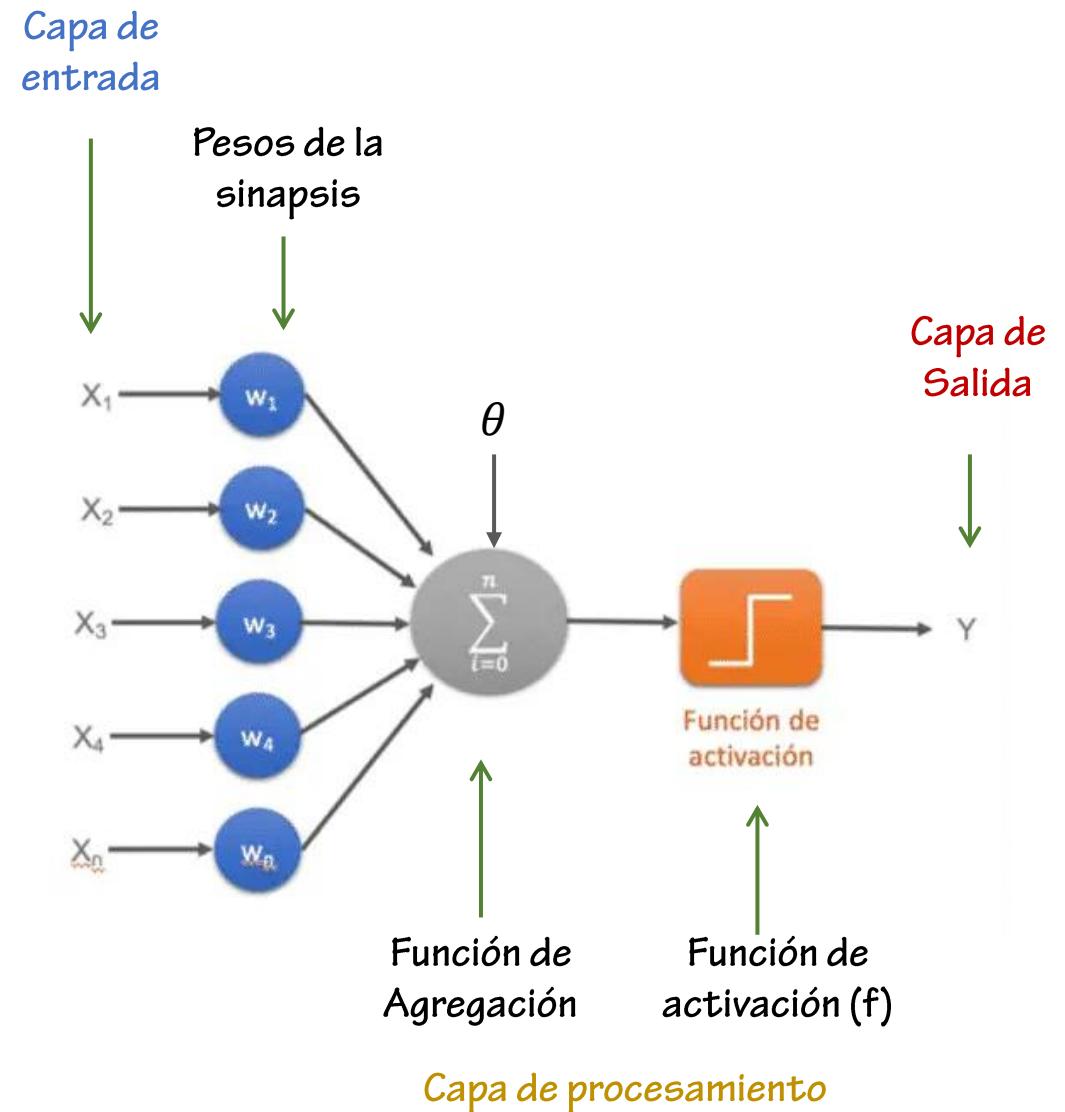
PERCEPTRÓN SIMPLE:

- Matemáticamente:

$$y(x) = f \left(\sum_{i=1}^n x_i w_i + \theta \right)$$

El perceptrón simple sólo es capaz de realizar separaciones lineales.

$$y(x) = \begin{cases} 0, & \text{Si } W \cdot x + \theta \leq 0 \\ 1, & \text{Si } W \cdot x + \theta > 0 \end{cases}$$



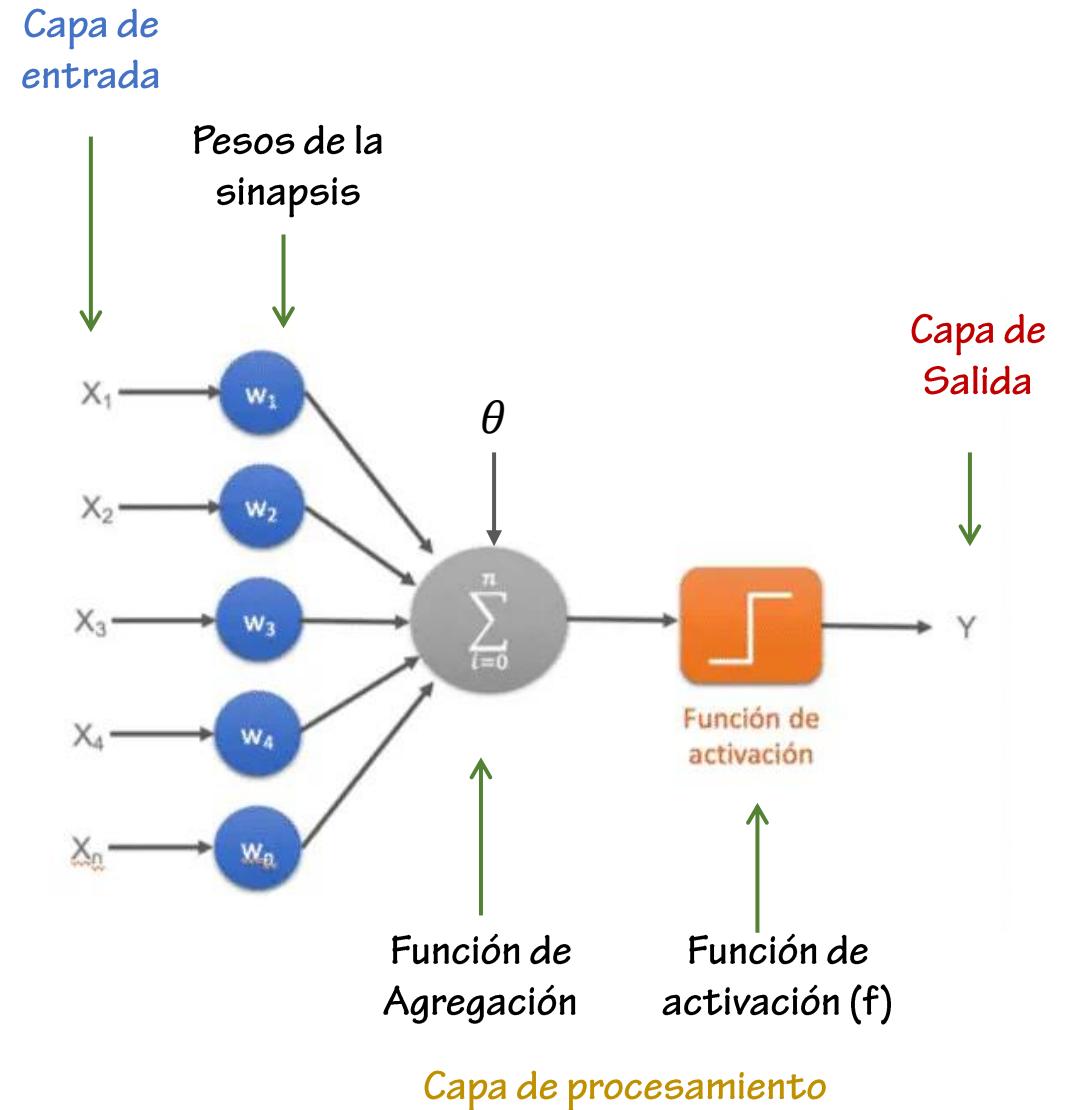
Redes Neuronales

PERCEPTRÓN SIMPLE - ENTRENAMIENTO:

El perceptrón **aprende de manera iterativa** siguiendo estos pasos:

1. Inicializar pesos y el umbral con valores aleatorios
2. Seleccione una instancia del conjunto de entrenamiento:

- 2.1 Determine la salida del perceptrón
- 2.2 Si hay un error en la salida se deben modificar los pesos w_i usando el factor: $\Delta w_i = \eta(y - \hat{y})x_i$ donde η es la tasa de aprendizaje. El umbral se debe modificar de la misma manera.
- 2.3 Si no se ha cumplido criterio de terminación, vuelva al paso 2.

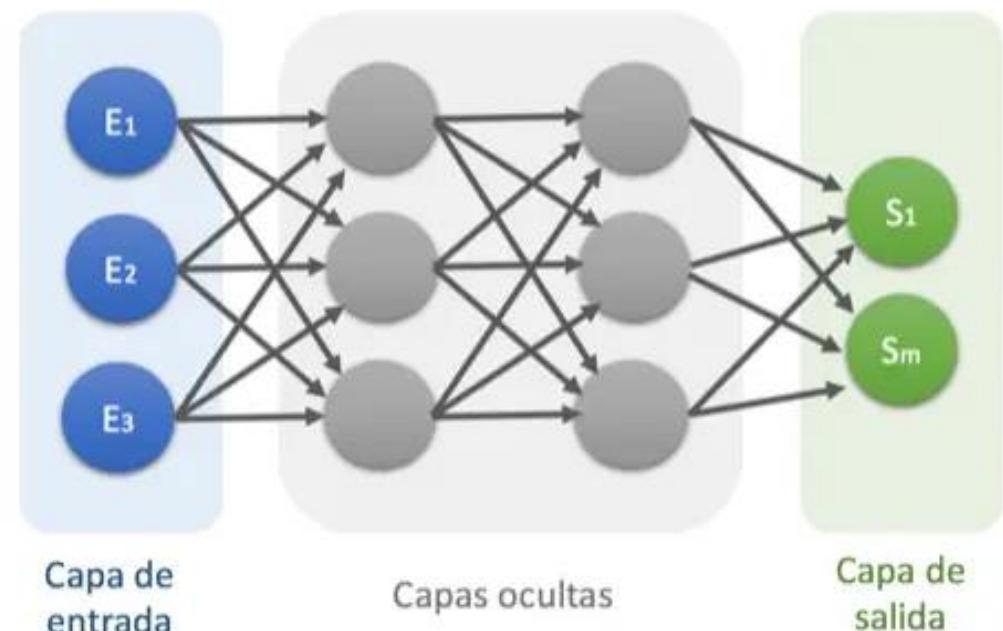


Redes Neuronales

PERCEPTRÓN MULTICAPA

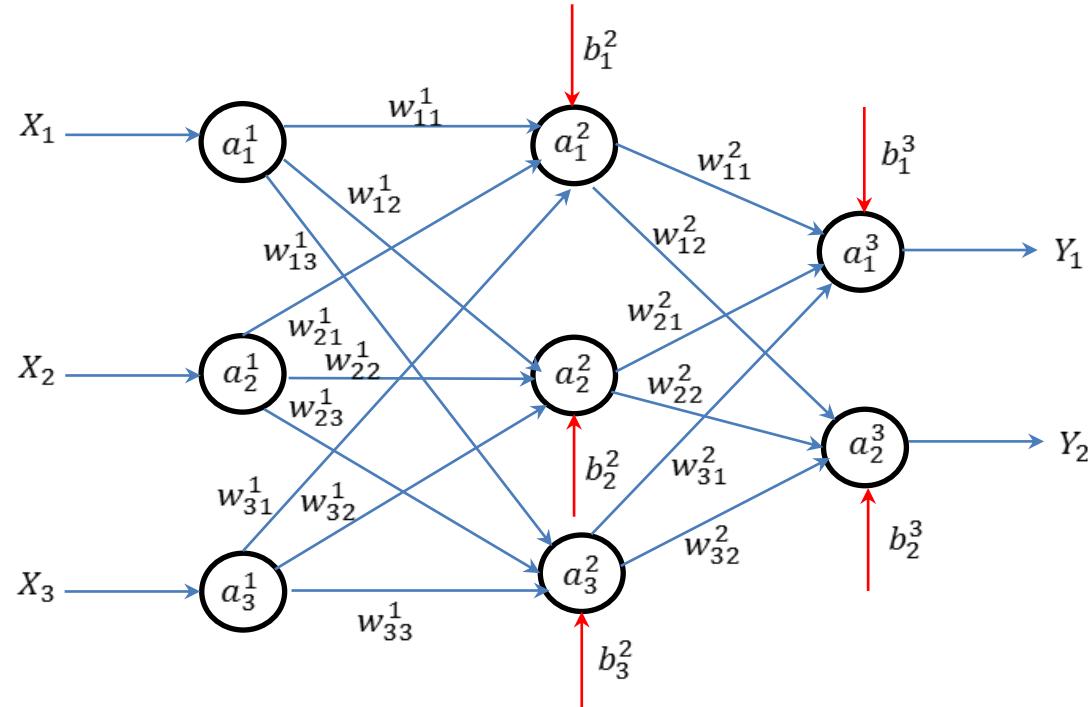
Es una red neuronal que está formada por múltiples capas de neuronas interconectadas que tienen la capacidad de resolver problemas que no son linealmente separables.

- **Capa de entrada:** conecta la red con el exterior, cada neurona se corresponde con cada una de las variables de entrada de la red.
- **Capas ocultas:** son una aglomeración de capas en las que cada activación de una salida procede de la suma ponderada de las activaciones de la capa anterior conectadas, mas sus correspondientes umbrales (bias, sesgos).
- **Capa de salida:** conecta las capas ocultas con la salida de la red que proporciona los resultados.



Redes Neuronales

PERCEPTRÓN MULTICAPA



- **C:** número de capas
- Capa de entrada = 1
- Capas ocultas = C-2
- Capas de salida = 1
- w_{ij}^q es el peso de conexión de la neurona i de la capa q con la neurona j de la capa q+1. con $q=1, 2, \dots, C-1$
- b_i^q es el umbral (o bias) de la neurona i de la capa q
- z_i^q es el valor pre activación de la neurona i de la capa q
- a_i^q es el valor post activación de la neurona i de la capa q
- N_q es el número de neuronas de la capa q
- W^q es la matriz de pesos entre la capa q y la capa q+1

Redes Neuronales

PERCEPTRÓN MULTICAPA – ACTIVACIÓN DE LAS CAPAS

Capa de Entrada:

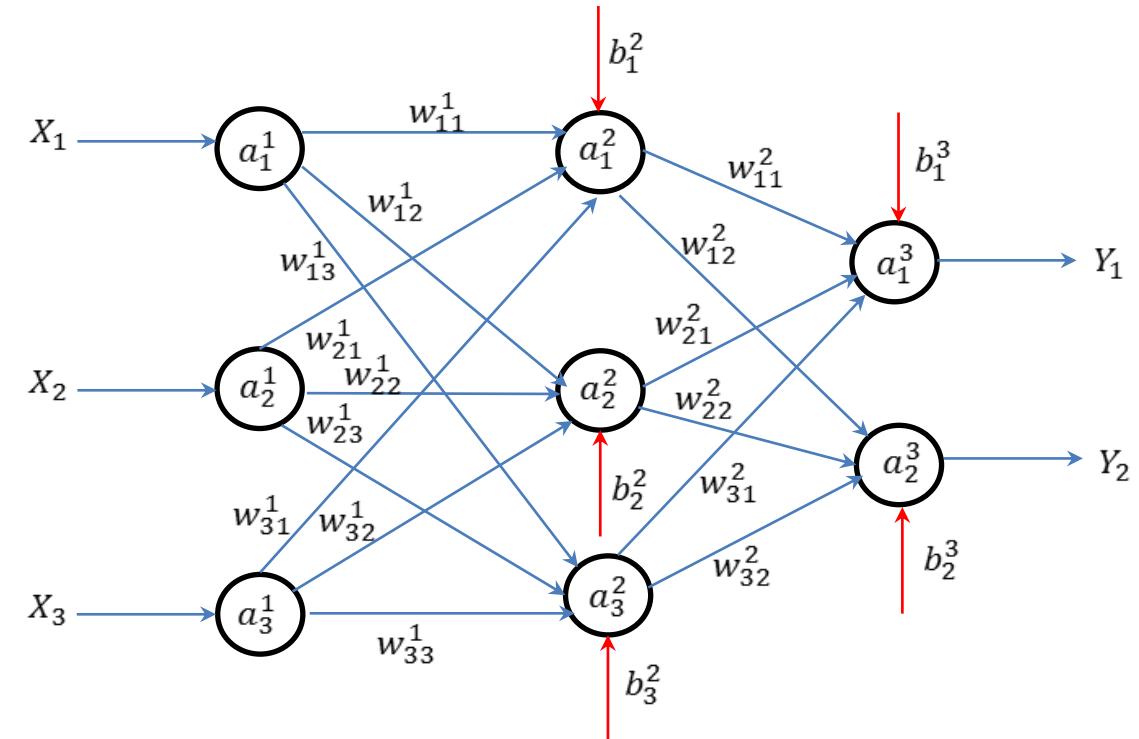
$$a_i^q = a_i^1 = x_i, \quad i = 1, 2, \dots, n$$

Capa de Salida:

$$a_i^c = y_i = f\left(\sum_{j=1}^{N_{(c-1)}} W_{ji}^{c-1} a_j^{c-1} + b_i^c\right), \quad i = 1, 2, \dots, N_c$$

Capas Ocultas:

$$a_i^q = f\left(\sum_{j=1}^{N_{(q-1)}} W_{ji}^{q-1} a_j^{q-1} + b_i^q\right), \quad i = 1, 2, \dots, N_q, q = 2, 3, \dots, C - 1$$

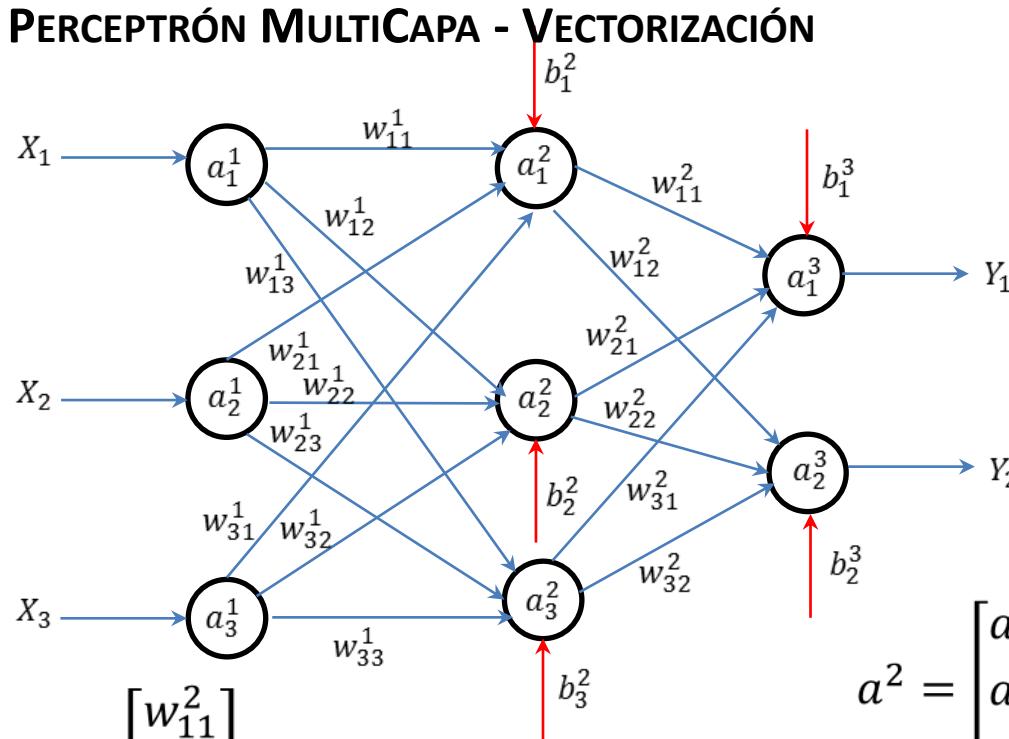


$$a_1^3 = f(w_{11}^2 * a_1^2 + w_{21}^2 * a_2^2 + w_{31}^2 * a_3^2 + b_1^3)$$

Redes Neuronales



PERCEPTRÓN MULTICAPA - VECTORIZACIÓN



$$w_{:,1}^2 = \begin{bmatrix} w_{11}^2 \\ w_{21}^2 \\ w_{31}^2 \end{bmatrix}$$

$$w_{:,2}^2 = \begin{bmatrix} w_{12}^2 \\ w_{22}^2 \\ w_{32}^2 \end{bmatrix}$$

$$W^2 = \begin{bmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \\ w_{31}^2 & w_{32}^2 \end{bmatrix}_{3x2}^{N_2 \times N_3}$$

$$a^2 = \begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix}_{3x1}^{N_2 \times 1}$$

$$b^3 = \begin{bmatrix} b_1^3 \\ b_2^3 \end{bmatrix}_{2x1}^{N_3 \times 1}$$

$$a_1^3 = f(z_1^3) = f(w_{11}^2 * a_1^2 + w_{21}^2 * a_2^2 + w_{31}^2 * a_3^2 + b_1^3)$$

$$a_2^3 = f(z_2^3) = f(w_{12}^2 * a_1^2 + w_{22}^2 * a_2^2 + w_{32}^2 * a_3^2 + b_2^3)$$

$$z^3 = \begin{bmatrix} z_1^3 \\ z_2^3 \end{bmatrix} = \begin{bmatrix} w_{11}^2 & w_{21}^2 & w_{31}^2 \\ w_{12}^2 & w_{22}^2 & w_{32}^2 \end{bmatrix}_{2x3}^{N_3 \times 1} \begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix}_{3x1}^{N_2 \times 1} + \begin{bmatrix} b_1^3 \\ b_2^3 \end{bmatrix}_{2x1}^{N_3 \times 1}$$

$$z^3 = W^2 T a^2 + b^3$$

$$a^3 = \begin{bmatrix} a_1^3 \\ a_2^3 \end{bmatrix} = f \left(\begin{bmatrix} z_1^3 \\ z_2^3 \end{bmatrix} \right) = f^3(z^3) = y$$

Redes Neuronales

PERCEPTRÓN MULTICAPA - VECTORIZACIÓN

Capa 2

$$z^2 = W^{1T} a^1 + b^2$$

$$a^2 = f^2(z^2)$$

Capa 3

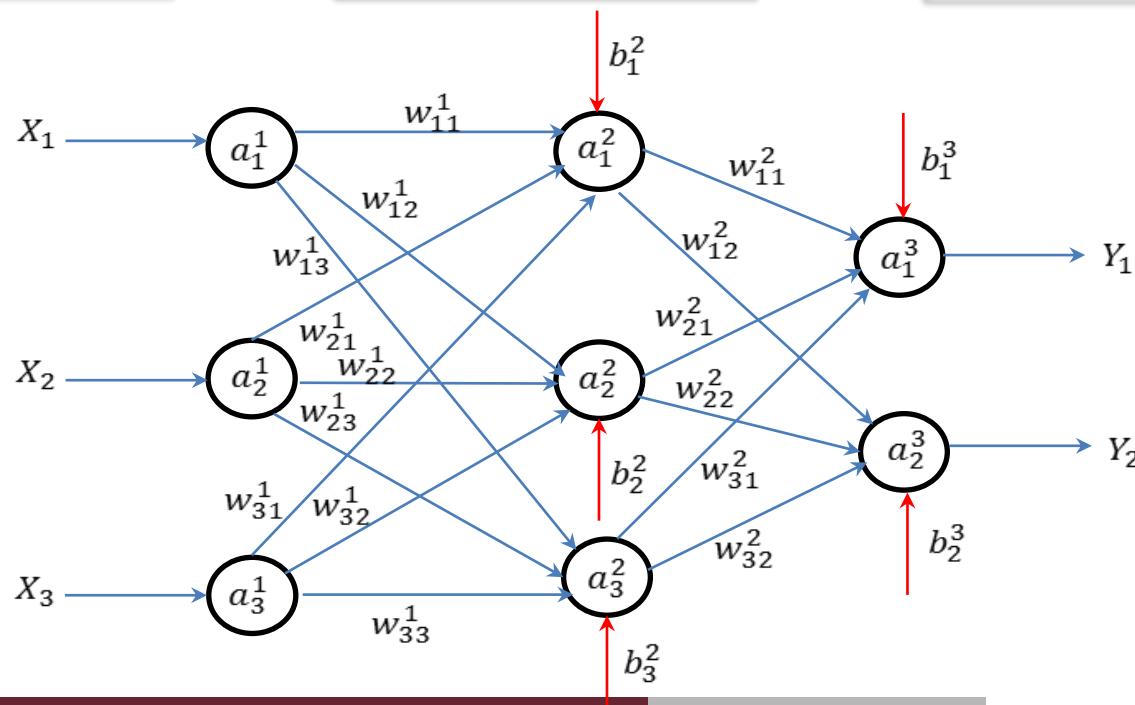
$$z^3 = W^{2T} a^2 + b^3$$

$$a^3 = f^3(z^3)$$

Capa *l* (cualquier Capa)

$$z^l = W^{l-1T} a^{l-1} + b^l$$

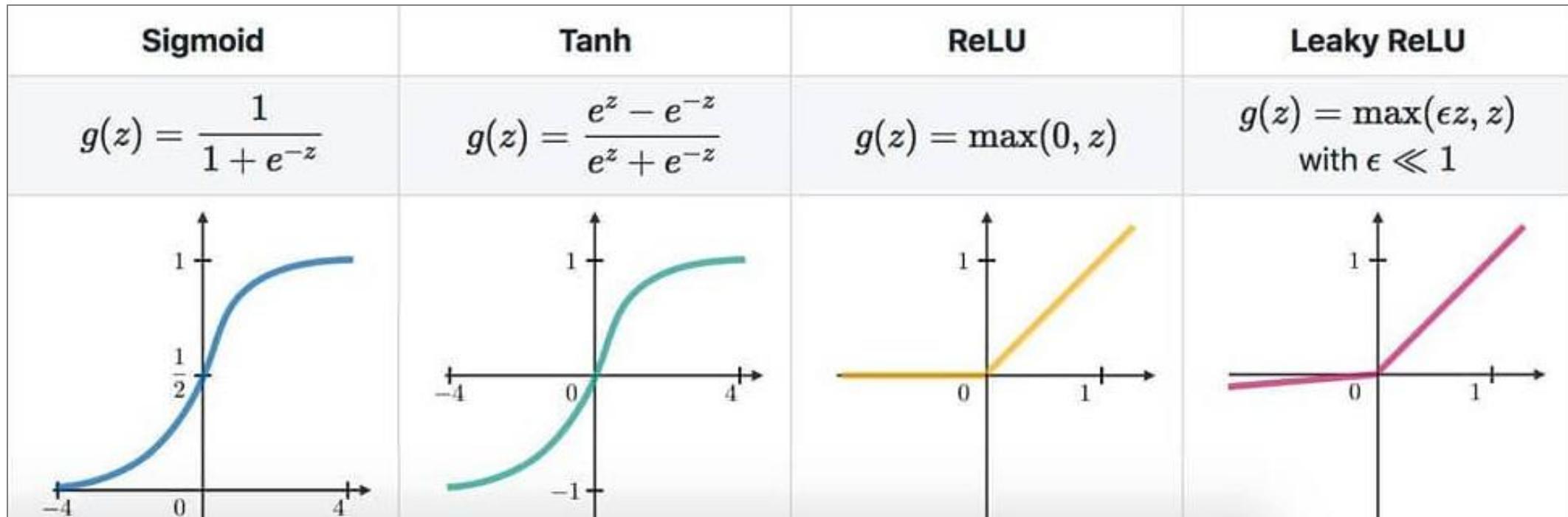
$$a^l = f^l(z^l)$$



Redes Neuronales

PERCEPTRÓN MULTICAPA – FUNCIONES DE ACTIVACIÓN

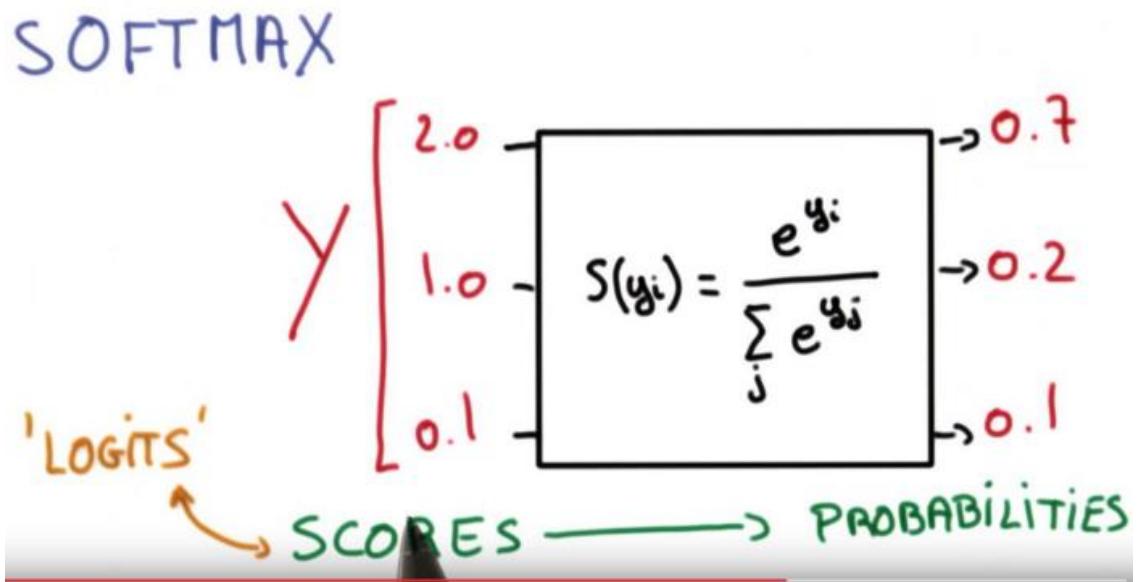
Es necesario aplicar una función de activación no lineal para hacer que la red sea más poderosa y pueda representar funciones complejas no lineal entre la entrada y la salida.



Redes Neuronales

PERCEPTRÓN MULTICAPA – FUNCIONES DE ACTIVACIÓN

Es necesario aplicar una función de activación no lineal para hacer que la red sea más poderosa y pueda representar funciones complejas no lineales entre la entrada y la salida.



La función Softmax transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas es 1.

Redes Neuronales



PERCEPTRÓN MULTICAPA – MÉTRICAS DE ERROR

Con el fin de ajustar los pesos de la red y los umbrales se debe calcular el error de la red neuronal.

Median Squared Error

$$\mathcal{L}(S, Y) = e(n) = \frac{1}{2} (S - Y)^2$$

$S(n)$: Salidas deseadas de la red para el patrón n

$Y(n)$: Vector de salida de la red para el patrón n

Logistic Regression Loss Function

$$\mathcal{L}(S, Y) = e(n) = -(S \log Y + (1 - S) \log(1 - Y))$$

Cost Function

$$J(\mathbf{w}, \mathbf{b}) = E = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(S, Y) = \frac{1}{N} \sum_{n=1}^N e(n)$$



Redes Neuronales

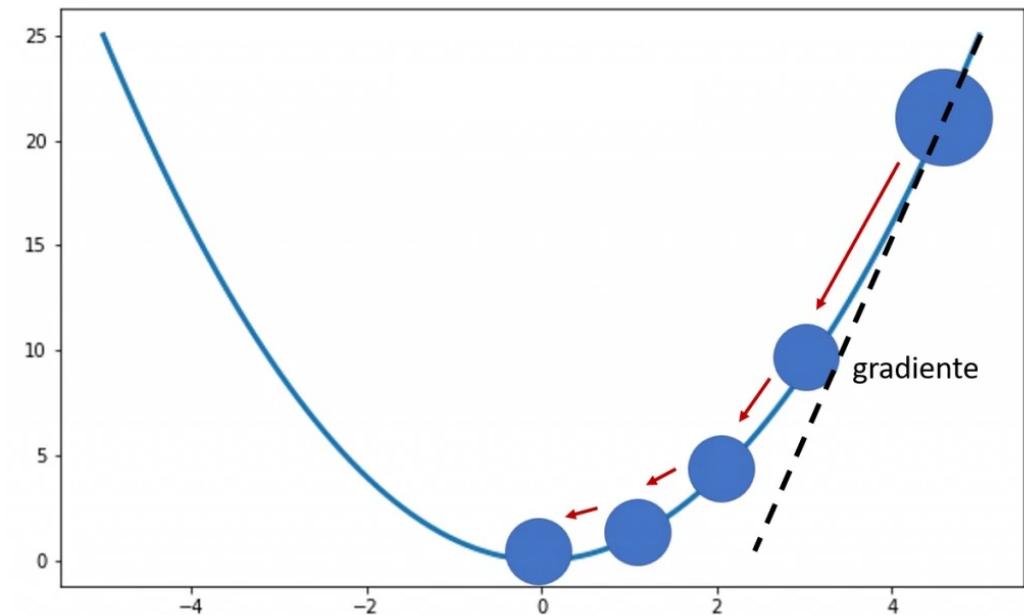
PERCEPTRÓN MULTICAPA – AJUSTE DE LOS PESOS Y LOS UMBRALES

Una vez se tiene el error se debe hacer la corrección de los pesos y de los umbrales de la última capa de la red. Para ello se usa el método de descenso del gradiente sobre los errores (lo que con un poco de matemática nos lleva a usar la derivada de la función de activación).

$$\omega_i = \omega_i + \Delta\omega_i$$

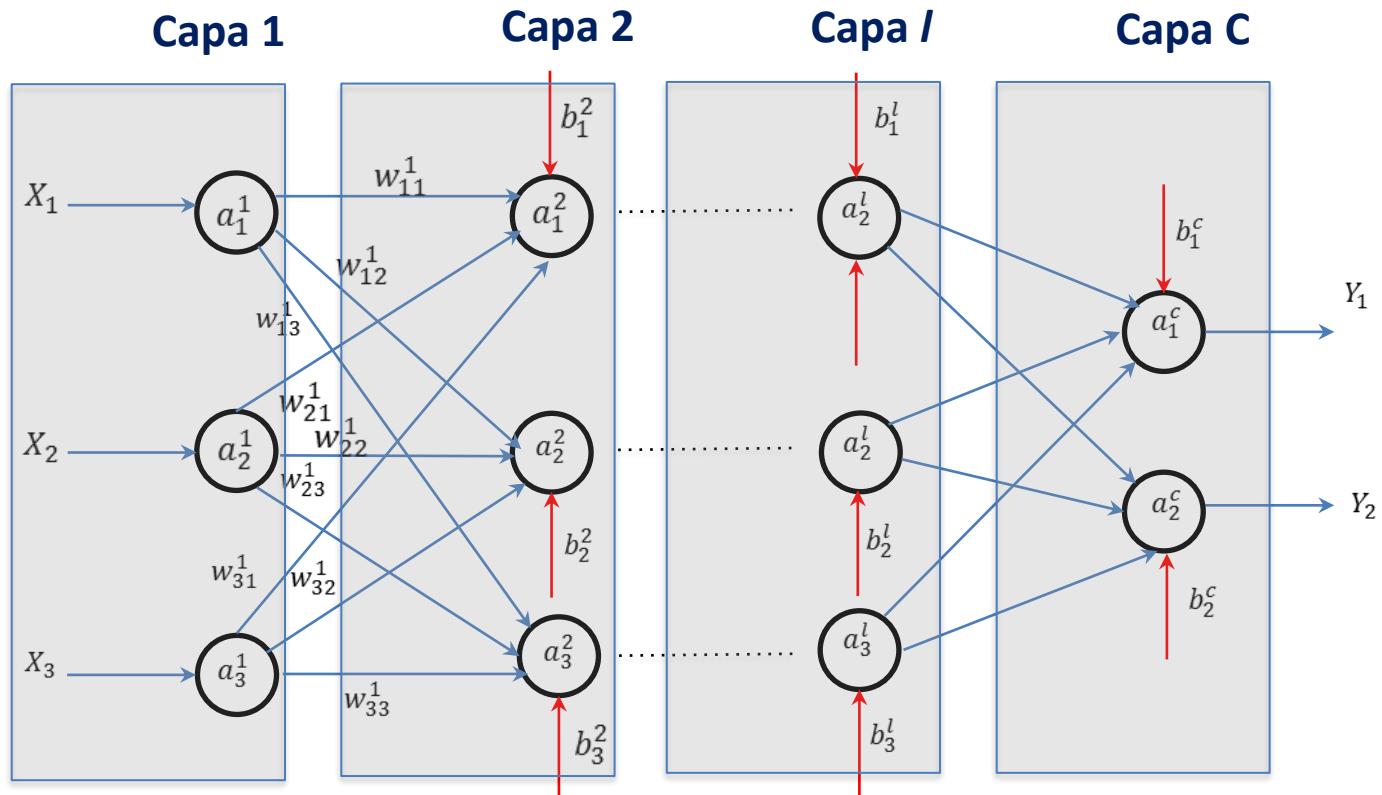
$$\Delta\omega_i = -\eta \frac{\partial e(w, b)}{\partial w} \quad \text{donde } \eta \text{ es la tasa de aprendizaje}$$

¿Qué pasa con los pesos y umbrales de las otras capas?



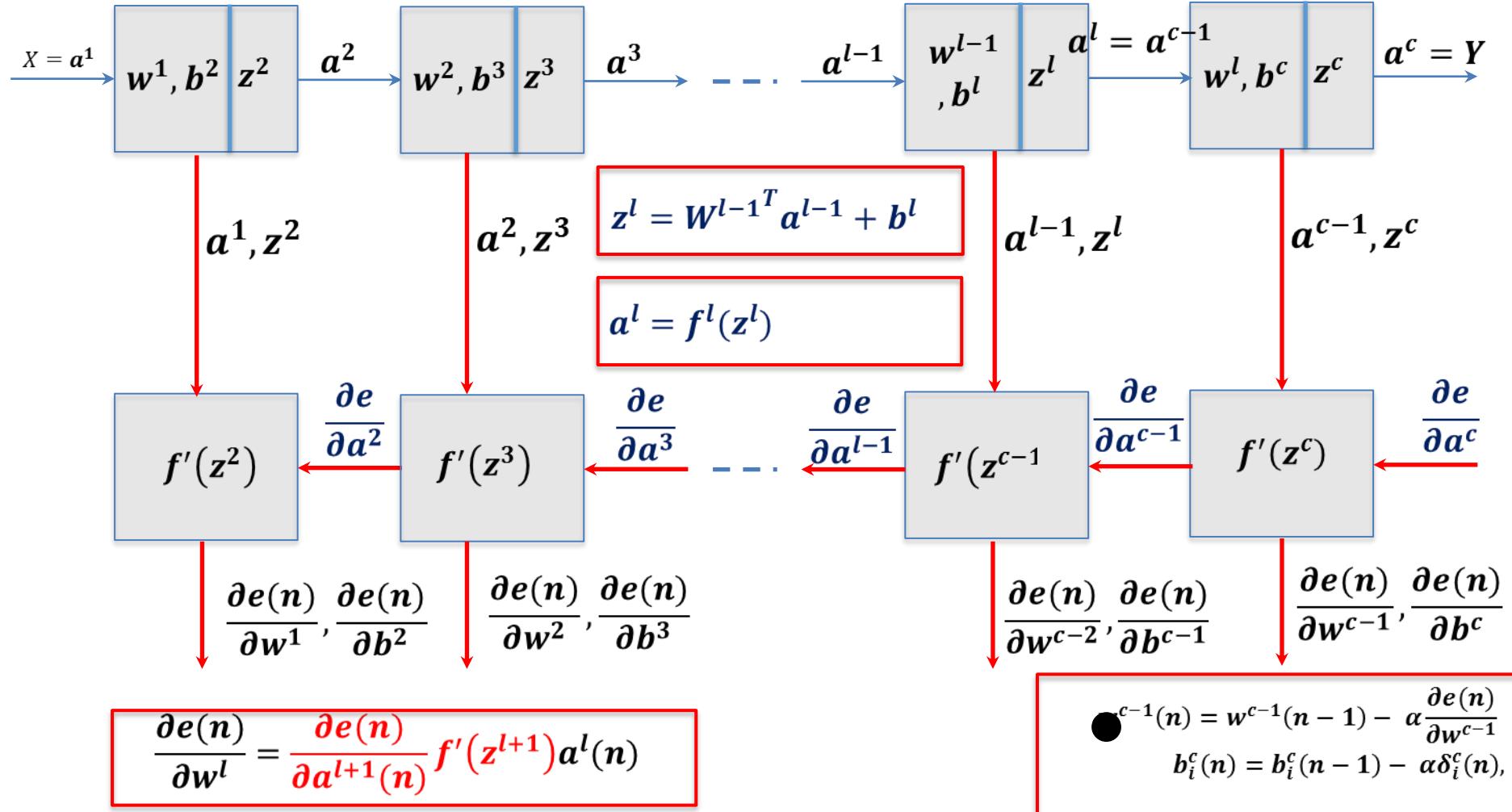
Redes Neuronales

PERCEPTRÓN MULTICAPA – PROPAGACIÓN HACIA ADELANTE (FORWARD)



Redes Neuronales

PERCEPTRÓN MULTICAPA – PROPAGACIÓN HACIA ADELANTE Y HACIA ATRÁS (FORWARD - BACKWARD)



Redes Neuronales



VENTAJAS:

- Tienen una alta capacidad de crear modelos no lineales para separar datos con fronteras de decisión complejas.
- Es un modelo de clasificación robusto al ruido
- Suelen tener un buen desempeño en diferentes tipos de problemas



DESVENTAJAS:

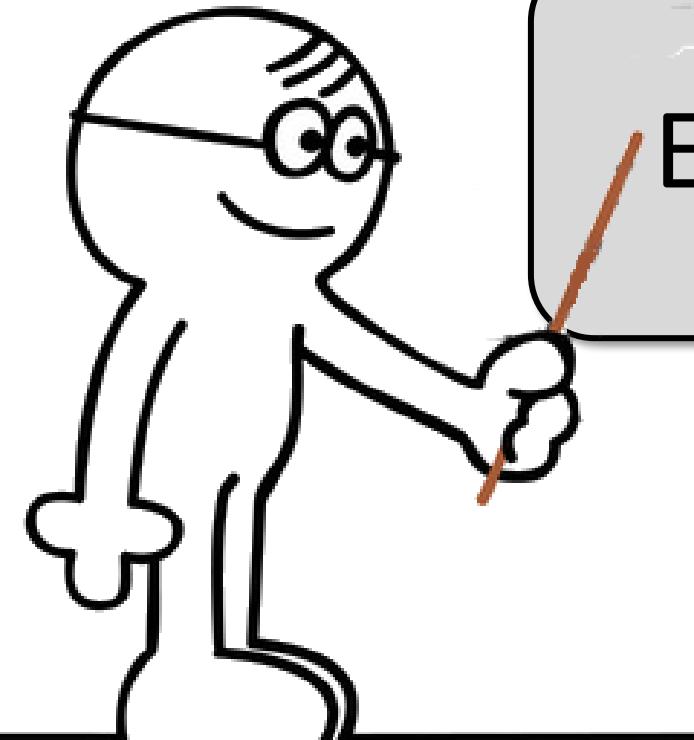
- Complejidad de aprendizaje para grandes tareas, cuanto más cosas se necesiten que aprenda una red, mas complicado será enseñarle.
- Tiempo de aprendizaje elevado. Esto depende de dos factores: primero si se incrementa la cantidad de patrones a identificar o clasificar y segundo si se requiere mayor flexibilidad o capacidad de adaptación
- No permite interpretar lo que se ha aprendido.
- Requiere muchos datos de entrenamiento
- Requiere el ajuste de muchos parámetros





VAMOS A CODIFICAR!

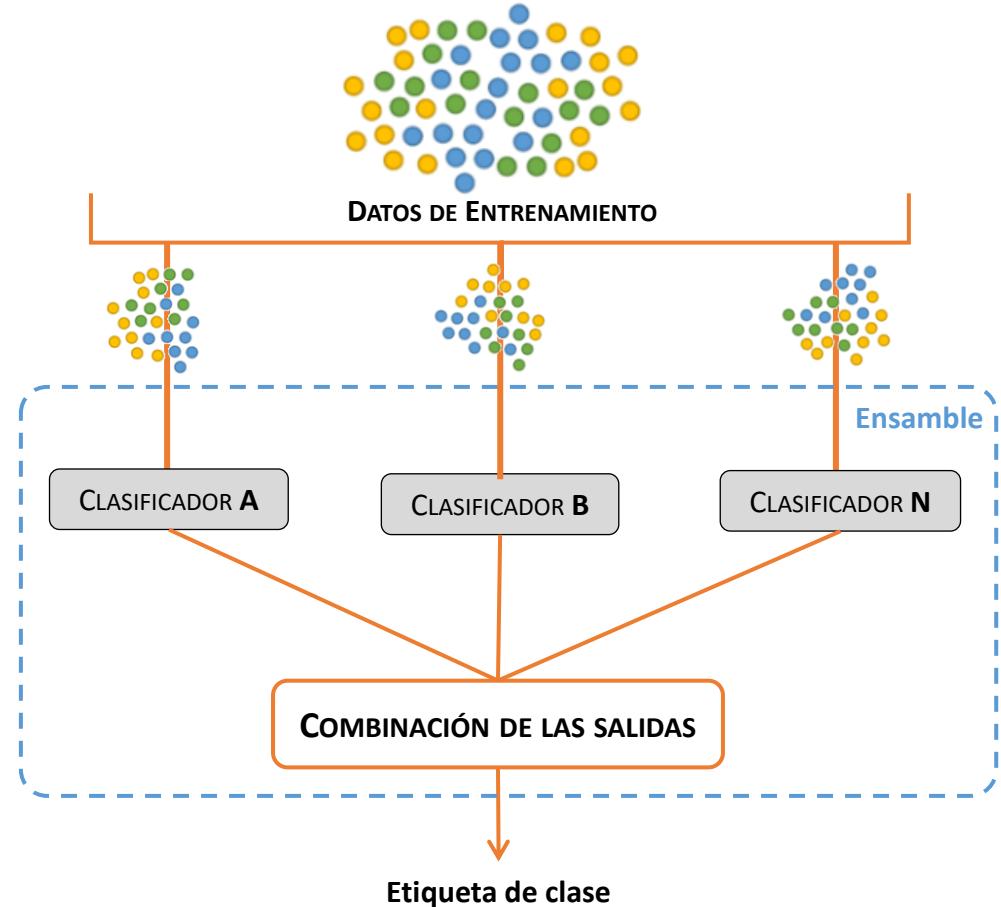
ENSAMBLES DE CLASIFICADORES



Ensambles de Clasificadores

MOTIVACIÓN

- Los clasificadores basados en ensambles intentan mejorar el rendimiento de los clasificadores individuales, combinándolos para obtener un nuevo clasificador que supere a cada uno de ellos.
- La idea básica es construir varios clasificadores a partir de los datos originales y luego combinar sus predicciones para clasificar los datos desconocidos.
- Esta idea sigue el comportamiento natural humano que tiende a buscar varias opiniones antes de tomar una decisión importante. Cuando las personas tienen que tomar decisiones difíciles, muchas veces toman en cuenta la opinión de varios expertos, buscando mejorar sus decisiones.



[Kuncheva2004] Ludmila Kuncheva. "Combining Pattern Classifiers, Methods and Algorithms"

[Rokach2009] L. Rokach *et al.* "Taxonomy for Characterizing ensemble methods in classification tasks"

Ensambles de Clasificadores



Importante:

Puede ser útil explotar la *diversidad entre clasificadores* y combinar sus salidas para mejorar el desempeño de los métodos tradicionales de clasificación [Kuncheva2004] [Rokach2009].

A partir de esta premisa surge la pregunta: ¿Cómo garantizamos la diversidad de los clasificadores?

BOOSTING

Ensambles de Clasificadores

BOOSTING

- Es una estrategia en la que el ensamble se construye de manera secuencial.
- El algoritmo funciona entrenando un clasificador inicial con un subconjunto aleatorio del conjunto de entrenamiento original.
- Los clasificadores posteriores se construyen ajustando los valores del error residual del clasificador anterior.
- Así, la idea general es tratar de centrar la atención de un clasificador en aquellas observaciones que el clasificador anterior estimó pobremente.
- Una vez que se crea la secuencia de los clasificadores, las predicciones hechas por los estos son ponderadas por sus puntuaciones de precisión y los resultados se combinan para crear una estimación final.
- Algunos de los modelos que normalmente se utilizan en la técnica de refuerzo son ADABOOST y XGBoost

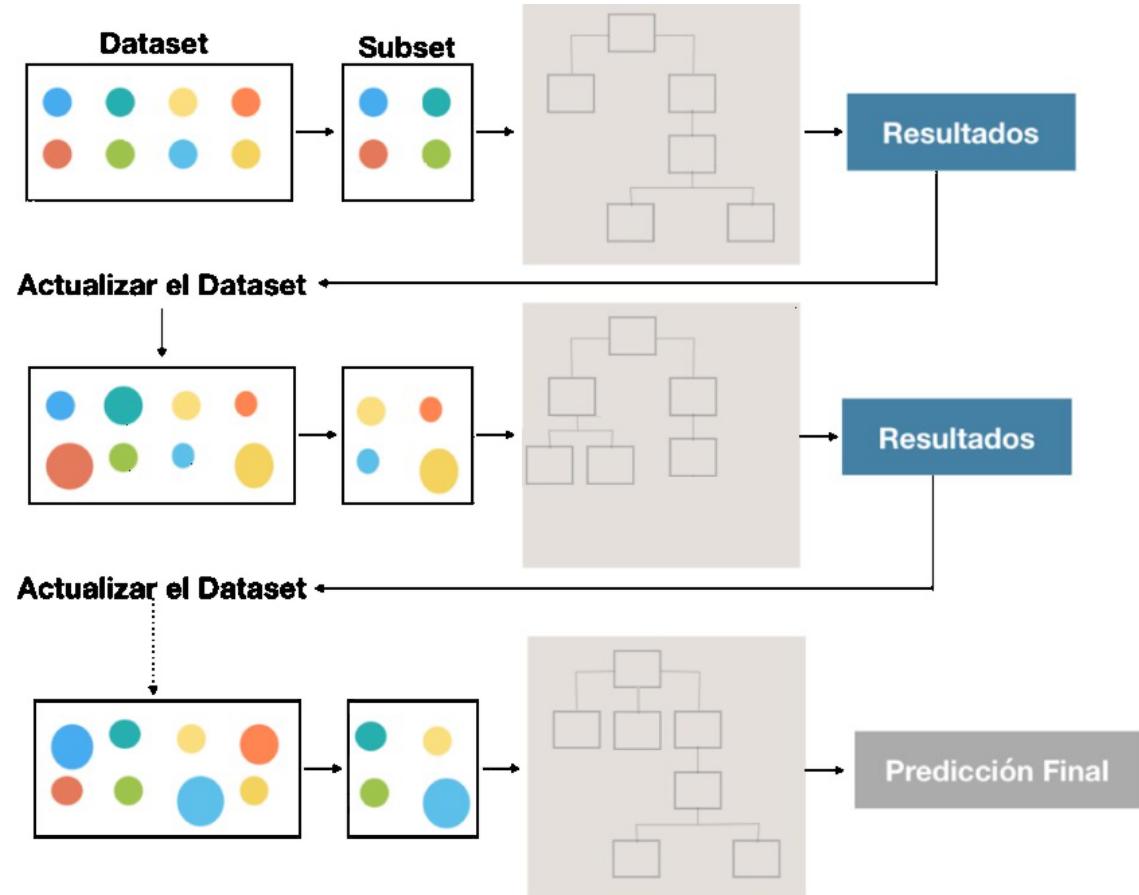


Imagen modificada de: <https://www.flickr.com/photos/ligdieli/48051633738/>

Ensambles de Clasificadores

BOOSTING: ADABOOST

AdaBoost

1. Assign every observation, x_i , an initial weight value,
 $w_i = \frac{1}{n}$, where n is the total number of observations.
2. Train a "weak" model. (most often a decision tree)
3. For each observation:
 - 3.1. If predicted incorrectly, w_i is increased
 - 3.2. If predicted correctly, w_i is decreased
4. Train a new weak model where observations with greater weights are given more priority.
5. Repeat steps 3 and 4 until observations are perfectly predicted or a preset number of trees are trained.

Chris Albon

https://chrisalbon.com/machine_learning/trees_and_forests/adaboost_classifier/

BAGGING

Ensambles de Clasificadores

BAGGING

- El **bagging** es utilizado para generar subconjuntos de datos haciendo una **selección aleatoria con reemplazo**, es decir, cada dato con el que se crea un subconjunto no deja de pertenecer al conjunto de datos original.
- Esto permite que cada observación pueda estar varias veces en el subconjunto o estar en diferentes subconjuntos.
- Cada subconjunto creado es utilizado para entrenar un clasificador independiente.
- La predicción final es el voto mayoritario entre todos los clasificadores entrenados.
- Uno de los algoritmos más usados de **Bagging** son los bosques aleatorios (**Random Forest**) basado en árboles de decisión.

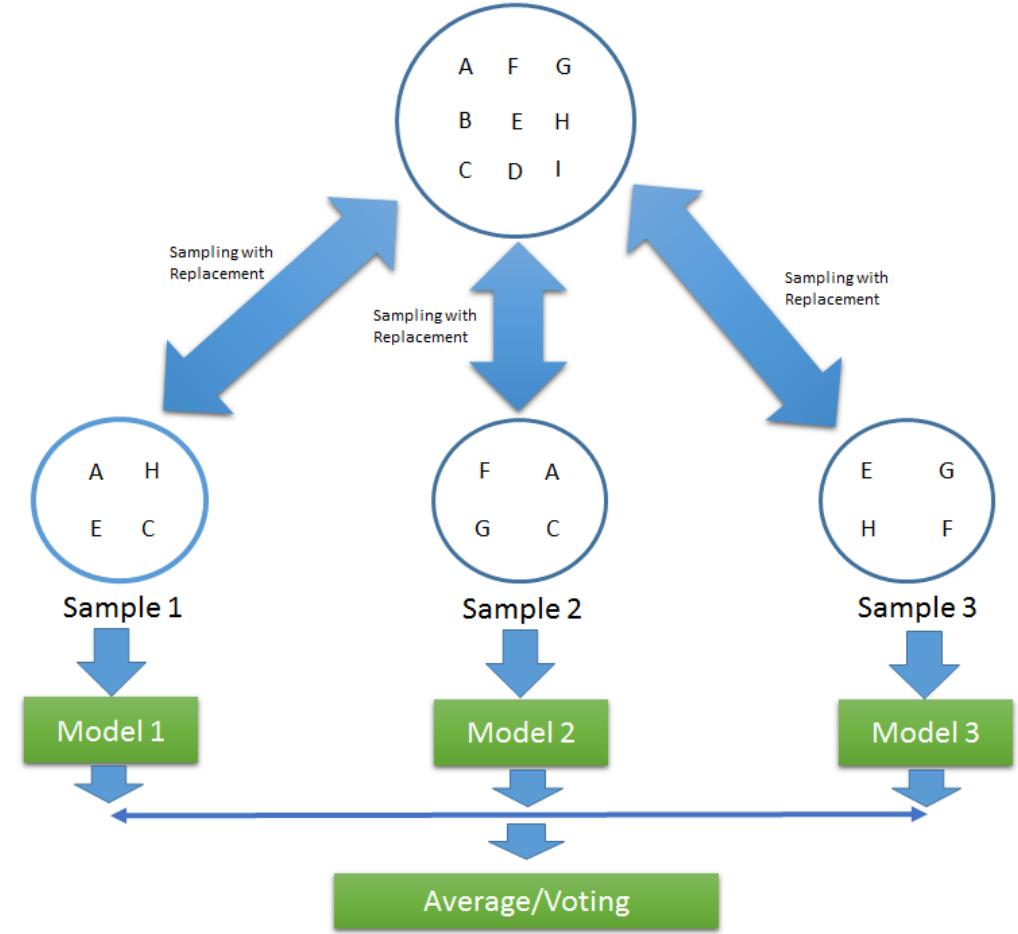


Imagen de: <https://medium.com/ml-research-lab/bagging-ensemble-meta-algorithm-for-reducing-variance-c98ffffa5489f>

SUB-ESPACIOS ALEATORIOS

Ensambles de Clasificadores

Sub-Espacios Aleatorios

- En este método cada modelo se entrena con todos los ejemplos, pero solo considera un **subconjunto de los atributos**. El tamaño de estos subconjuntos es el parámetro del método, y de nuevo el resultado es el promedio o votación de los resultados individuales de los modelos.

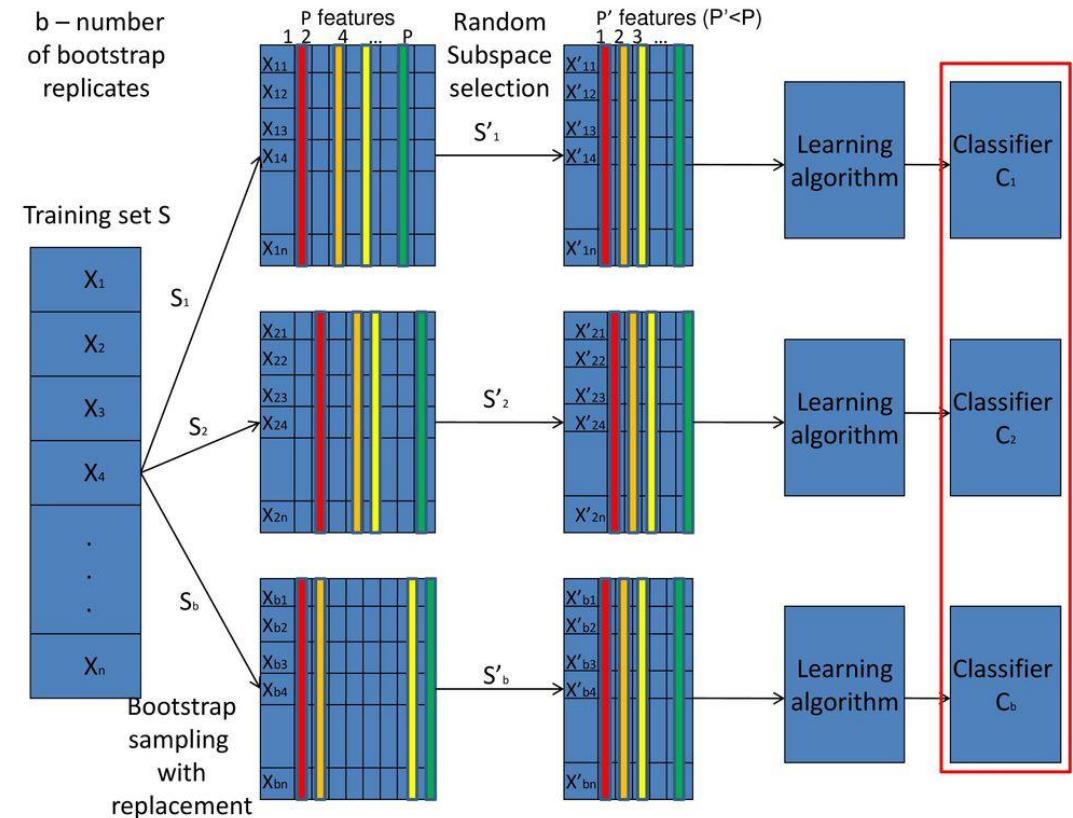


Imagen de: <https://medium.com/ml-research-lab/bagging-ensemble-meta-algorithm-for-reducing-variance-c98ffffa5489f>

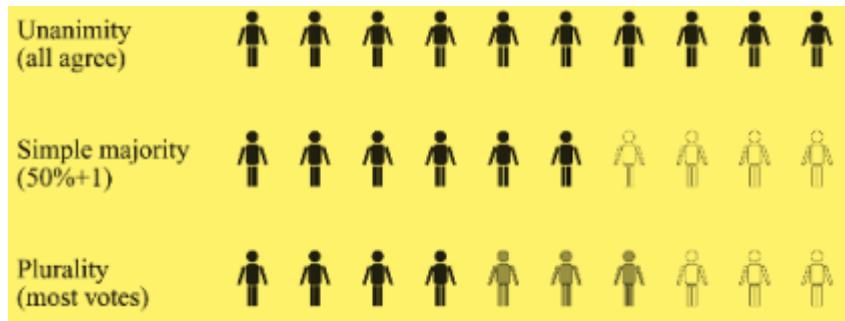
ESTRATEGIAS DE COMBINACIÓN

Ensambles de Clasificadores

DISEÑO DEL COMBINADOR:

Para métodos de clasificación:

- ✓ Voto Mayoritario
- ✓ Voto por Pluralidad
- ✓ Voto Ponderado



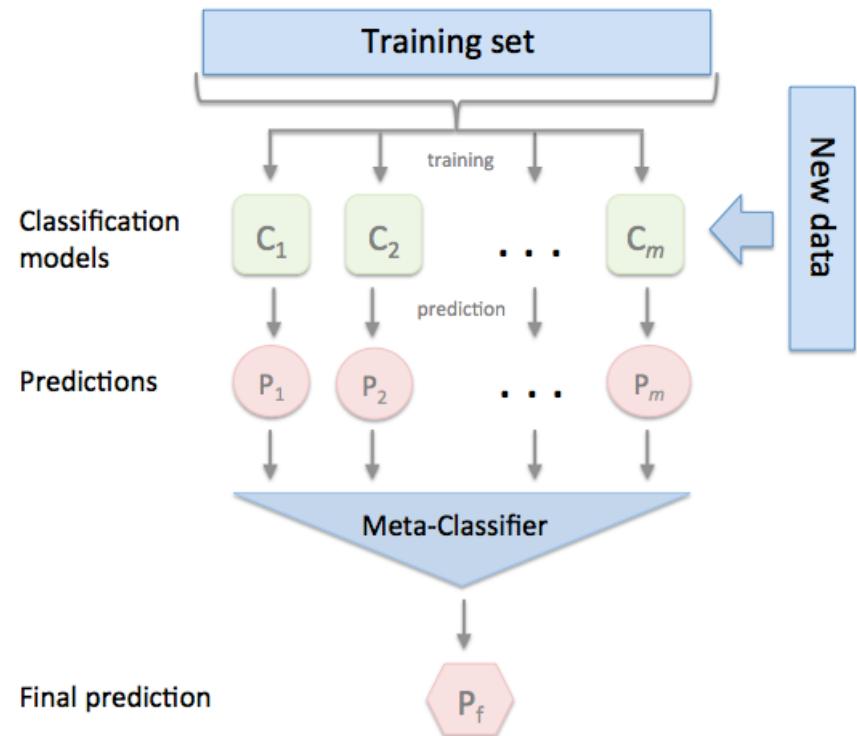
Para métodos de regresión:

- ✓ *Promedio Simple*
- ✓ *Promedio Ponderado*
- ✓ Reglas del Máximo y el Mínimo

$$H(x) = \frac{1}{L} \sum_{i=1}^L h_i(x)$$

Combinación por Stacking:

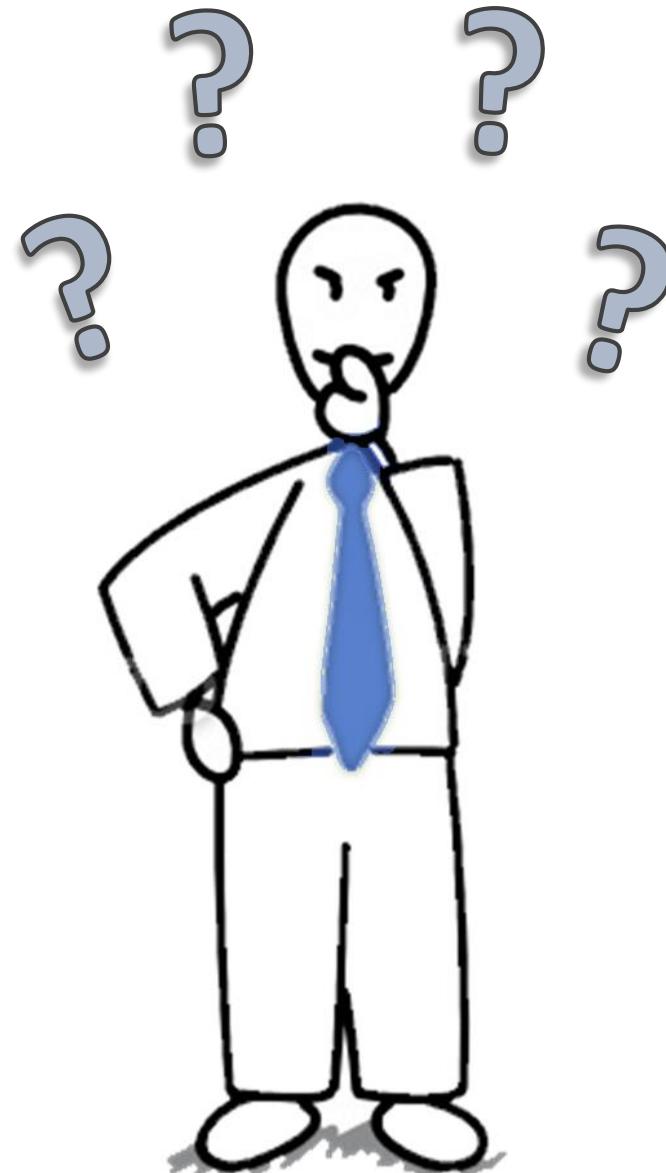
- ✓ Con las salidas de los clasificadores individuales se entrena otro clasificador





VEAMOS UN EJEMPLO APLICADO A UN
PROBLEMA REAL

Preguntas ...



CARLOS ANDRÉS MERA BANGUERO, Ph.D.
camerab@unal.edu.co