

# Aprendizaje Supervisado

---

**Clasificación y regresión mediante  
aprendizaje supervisado**



Universidad  
**Antonio de Nebrija**

Constantino Malagón Luque  
[cmalagon@nebrija.es](mailto:cmalagon@nebrija.es)

# Agenda

---

- Introducción a lo que veremos (y lo que no) durante esta hora y media.
  - Uff, ¿hora y media?
- Modelos para la extracción del conocimiento mediante métodos de aprendizaje supervisado
  - Clasificación / Regresión
- Ejemplo de aplicación

# Lo que deberíamos saber al final...

---

- No mucho...
- Aunque habré oído hablar de algunos algoritmos que me servirán algún día de estos...
- Y me acordaré de esas charlas que me dieron durante dos días.

# Extracción del conocimiento

---

# Data mining como proceso

## Preparación de datos

Obtener un conjunto de datos en formato tabular y con atributos bien definidos.



## Construcción del modelo

Aplicación de una o más técnicas.



## Validación

Garantizar que el modelo tendrá un buen rendimiento.




## Aplicación

Generar resultados con el modelo.

# Fase 1: Preparación de los datos

---

- Esto lo vimos en la charla anterior.
- Menos mal porque es lo más tedioso 
- Y lo que más tiempo lleva casi siempre en los experimentos

# Datos

---

- Pero recordad que estamos con aprendizaje supervisado
- Así que lo que yo pido son tablas con todos los datos, y **deben estar etiquetados con su clase**

# Fase 2: Construcción del modelo

---

- En esta fase pasaremos la siguiente hora y veinte
- Y, ¿qué es lo que veremos?



# Modelos

---

- Nuestro objetivo es analizar los datos para extraer conocimiento.
- Este conocimiento puede ser en forma de reglas, relaciones, asociaciones, patrones o, en general, modelos.
- Pues esto es lo que veremos: diferentes técnicas para la construcción de modelos que me permitan clasificar / predecir.

# Construcción del modelo

---

- Nosotros veremos estos:
  - Árboles de decisión
  - Redes neuronales
  - Redes bayesianas
  - K-NN

# Construcción del modelo

---

- Pero hay muchos más:
  - Support vector machines
  - Aprendizaje por refuerzo
  - Aprendizaje basado en casos
  - Diferentes tipos de regresión

# Construcción del modelo

---

- Y no sólo eso, sino que se pueden combinar (ensemble of classifiers)
  - Combinadores homogéneos
    - Bagging
    - Boosting
  - Combinadores heterogéneos
    - Stacking
    - grading
- Veremos algunos de estos también.
- Al final resulta que esto no parece que sea tan fácil

# Un ejemplo de clasificación

---

# Ejemplo de aplicación

---

- Clasificación de rayos gamma en Astrofísica.
- Descripción del problema:
  - Cascadas electromagnéticas debido al choque con moléculas de la atmósfera
  - Se trata de clasificar aquellos fotones producidos por rayos gamma muy energéticos de los hadrones que se producen (background).
- Esta información permitirá inferir información del objeto que los ha producido

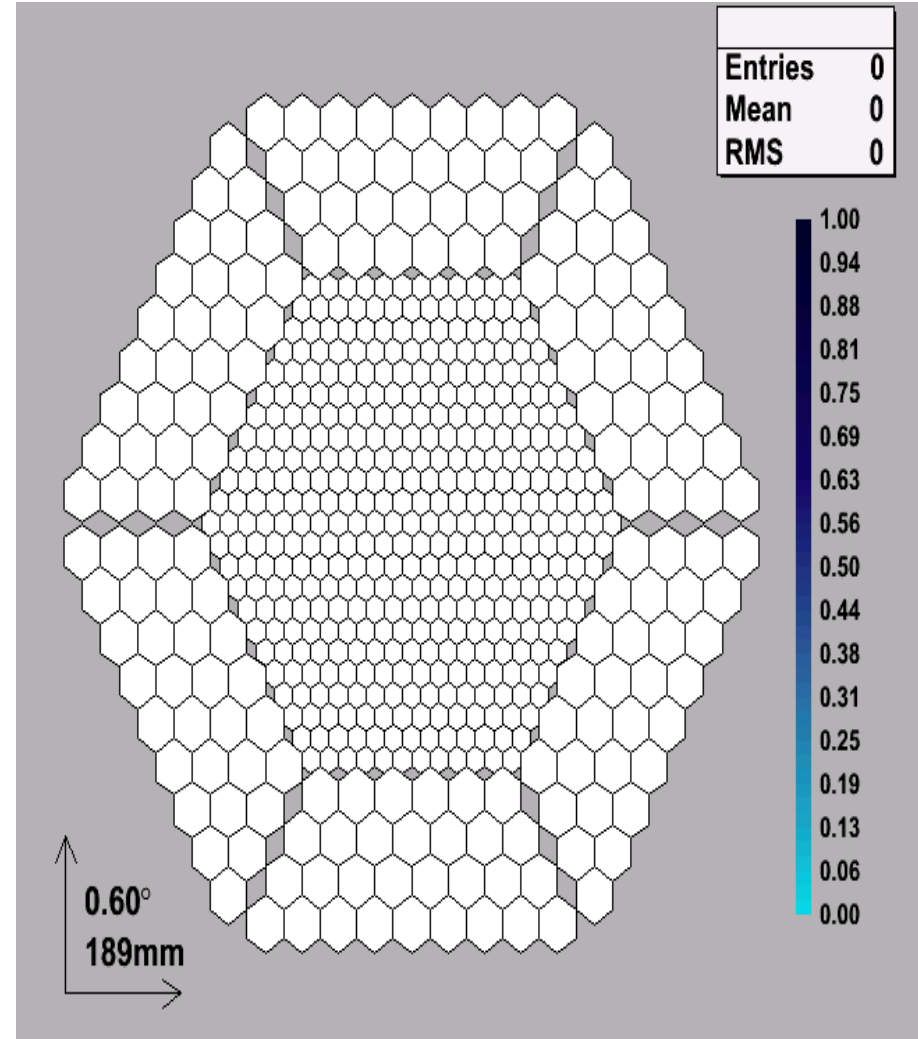
# Ejemplo de aplicación

---

- Los datos del experimento proceden del telescopio Cherenkov MAGIC situado en el observatorio de Roque de los Muchachos en la isla canaria de La Palma y perteneciente al Instituto Astrofísico de Canarias.



# Ejemplo de aplicación





# Ejemplo de aplicación

---

- Datos procedentes de simulación por métodos de Monte Carlo.
  - A falta de expertos...
- Los datos generados se transforman en una elipse, a partir de los cuales se obtienen los llamados parámetros de Hillas.
- Por ejemplo, longitud de los ejes mayor y menor de la elipse generada en la cámara, ángulo cenital, distancia al origen, etc.)

# Ejemplo de aplicación

---

- Cojamos por simplicidad dos de esos atributos: el eje mayor y el eje menor de la elipse.
- Con estos datos intentaremos construir un sistema clasificador.

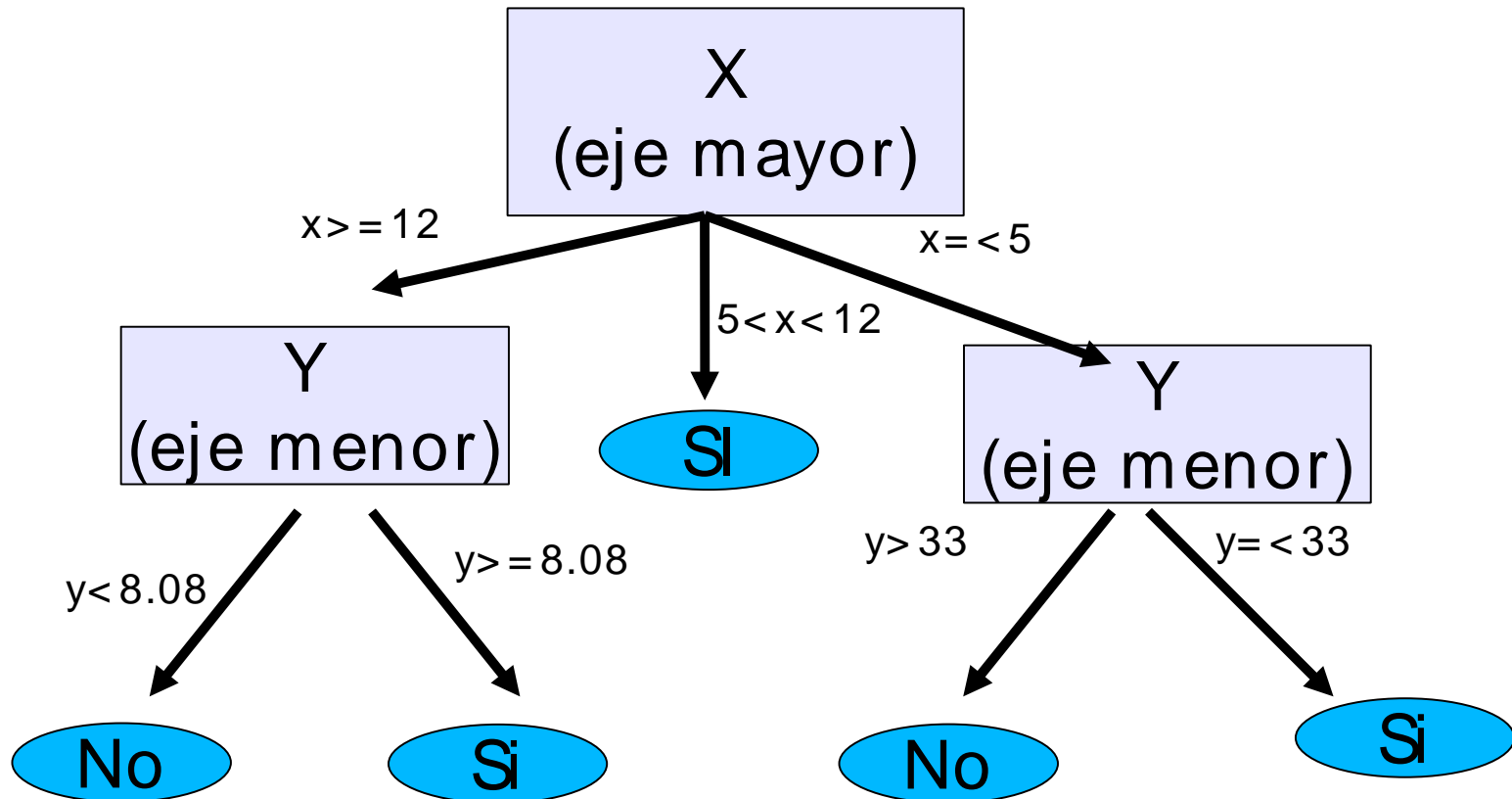
# Ejemplo de aplicación

- Veamos cómo son nuestros datos:

Eje mayor elipse (X)	Eje menor elipse (Y)	Gamma
19,119	11,978	Sí
88,709	8,637	Sí
63,100	21,900	Sí
48,050	13,550	Sí
69,989	18,122	No
22,777	12,708	No
121,603	59,445	No

# Árboles de decisión

- Esto es lo que buscamos.

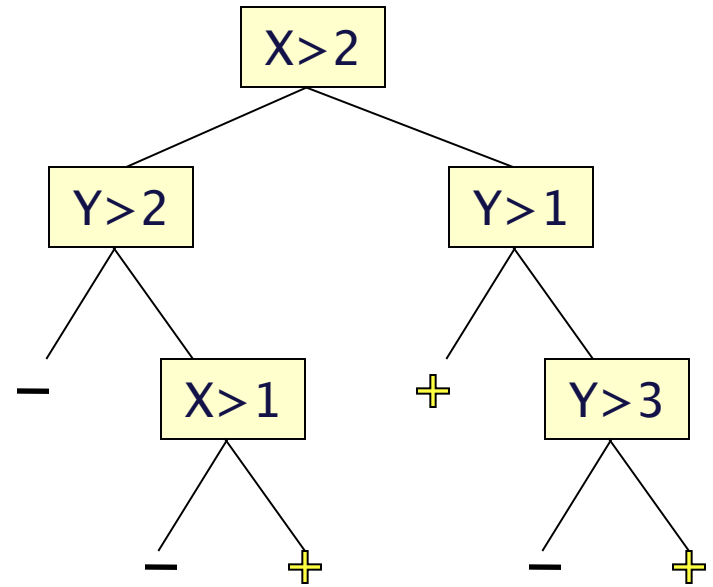
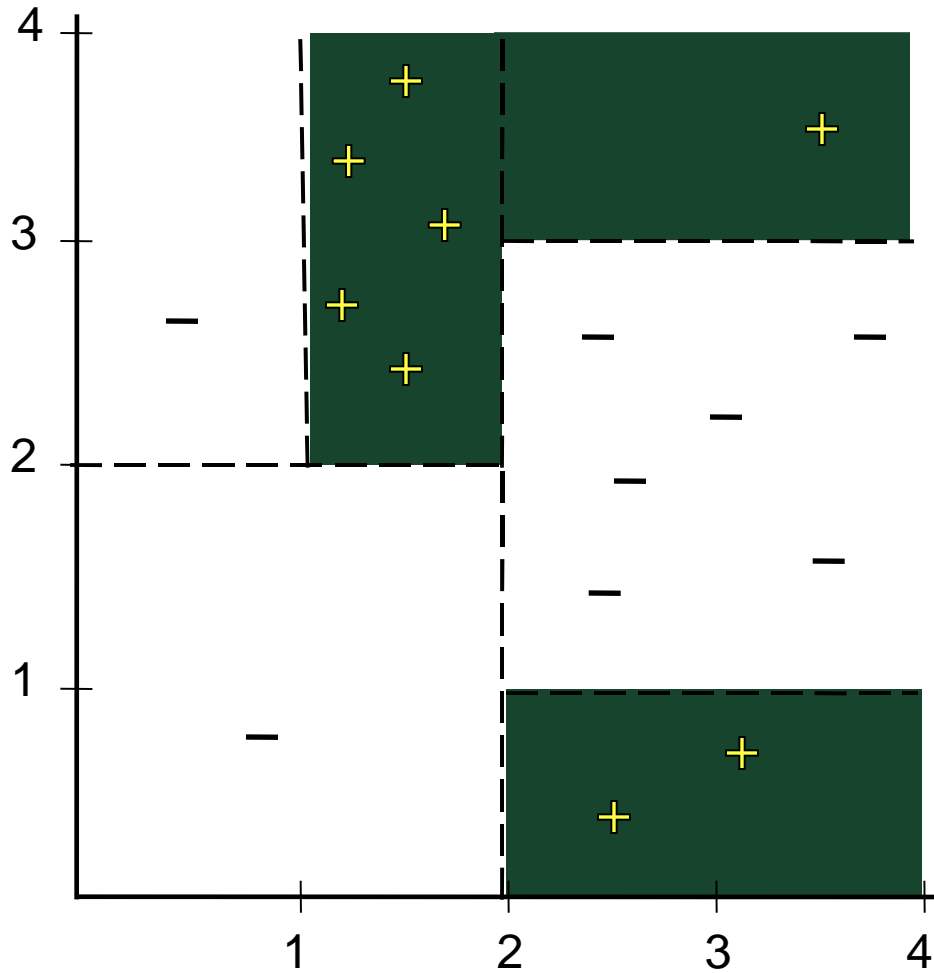


# Algoritmo ID3

---

- Propuesto por Ross Quinlan (Quinlan., 1986)
- Se evalúa cada atributo para elegir el más representativo para nuestro conjunto de datos.
  - Criterio de la entropía de Shannon (ya visto)
- Tendremos descripciones conjuntivas según vamos bajando en cada rama.

# Regiones de decisión



Regiones de decisión paralelas a los ejes y en forma de “escalón”.

# Regresión con árboles

---

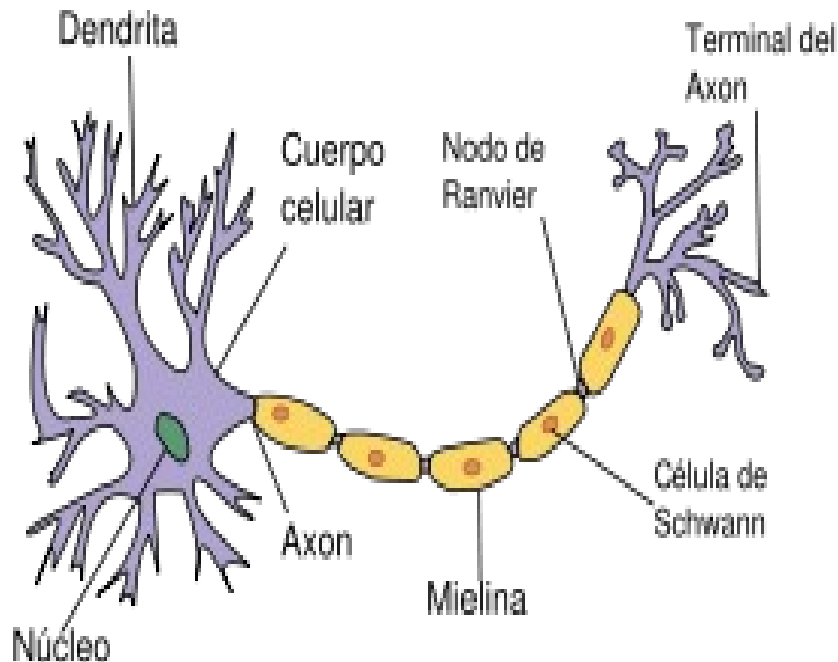
- **Árboles de regresión:** predicen la media de los valores de una hoja.
- **CART**, (Breiman et al., 1984).

# Redes neuronales artificiales

---



# La neurona biológica

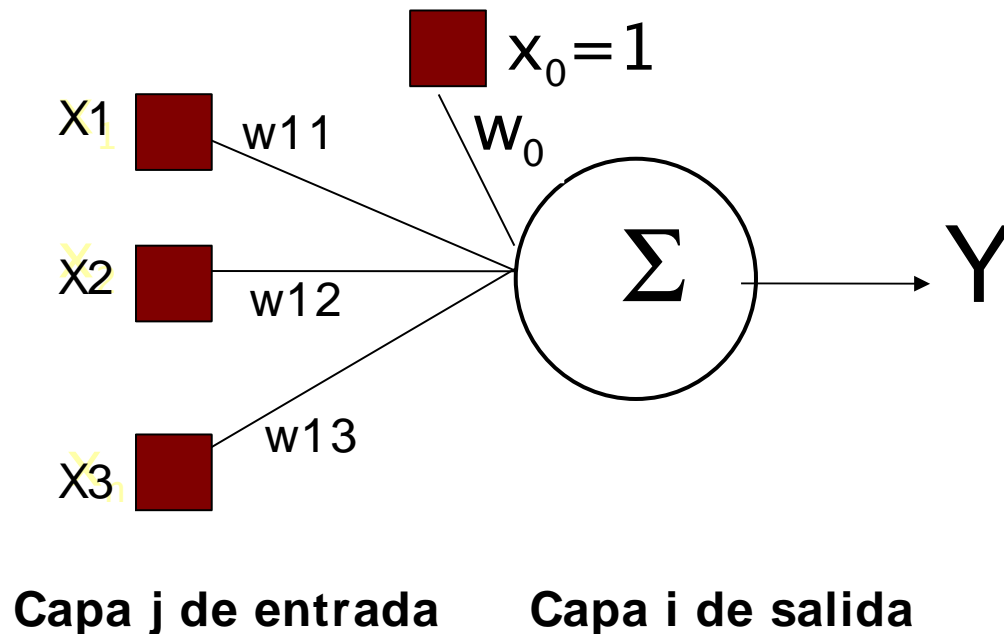


- Cargada electronegativamente:  $-70\text{meV}$
- Canales iónicos
- Si supera este umbral se produce una diferencia de potencial
- ¡Y tenemos información!

# La neurona artificial

- Linear Threshold Units (LTU)
- Similitud biológica

## Perceptrón simple



- Unidad de entrada
- Pesos  $w_{ij}$
- Capa de entrada
- Capa de salida

# El perceptrón simple: algoritmo

---

- Se presenta el primer patrón.
- ¿Qué le llega a la neurona de salida? Se calcula el  $\text{Netinput}_i$  como suma ponderada
  - $\text{Netinput}_i = w_0 + w_{11}x_1 + w_{12}x_2 + \dots + w_{1n}x_n$
- Se calcula la activación en la neurona  $i$  mediante una función de activación
  - lineal – umbral

$$\left\{ \begin{array}{ll} 1 & \text{Netinput}_i > 0 \\ 0 & \text{en caso contrario} \end{array} \right.$$
  - No lineal – sigmoide  $a_i = 1/(1 + \exp(-\text{Netinput}_i))$

# El perceptrón simple: algoritmo

---

- Se computa el error cometido:

$$\delta_i = (t_i - y_i)$$

“lo que debe salir menos lo que sale”

- Si el error es cero, ha clasificado bien
- Si no la red modifica sus pesos para que en la siguiente iteración ese error disminuya.

# El perceptrón simple: algoritmo

---

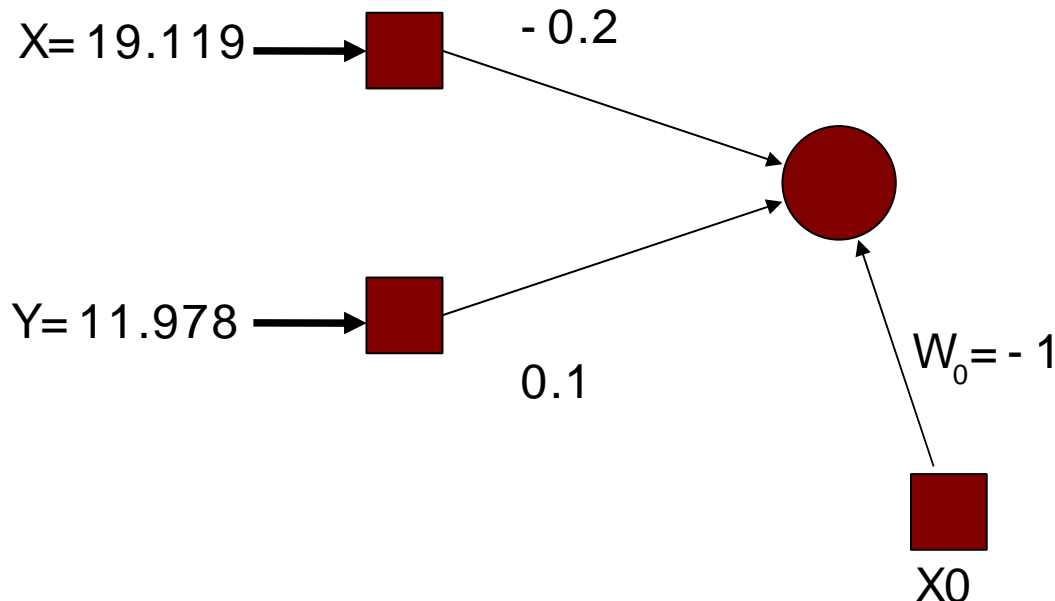
- Siempre de acuerdo a una regla de aprendizaje

$$\Delta w_{ij} = w_{ij}^f - w_{ij}^o = \epsilon x_j \delta_i$$

Esta es la famosa regla delta de aprendizaje

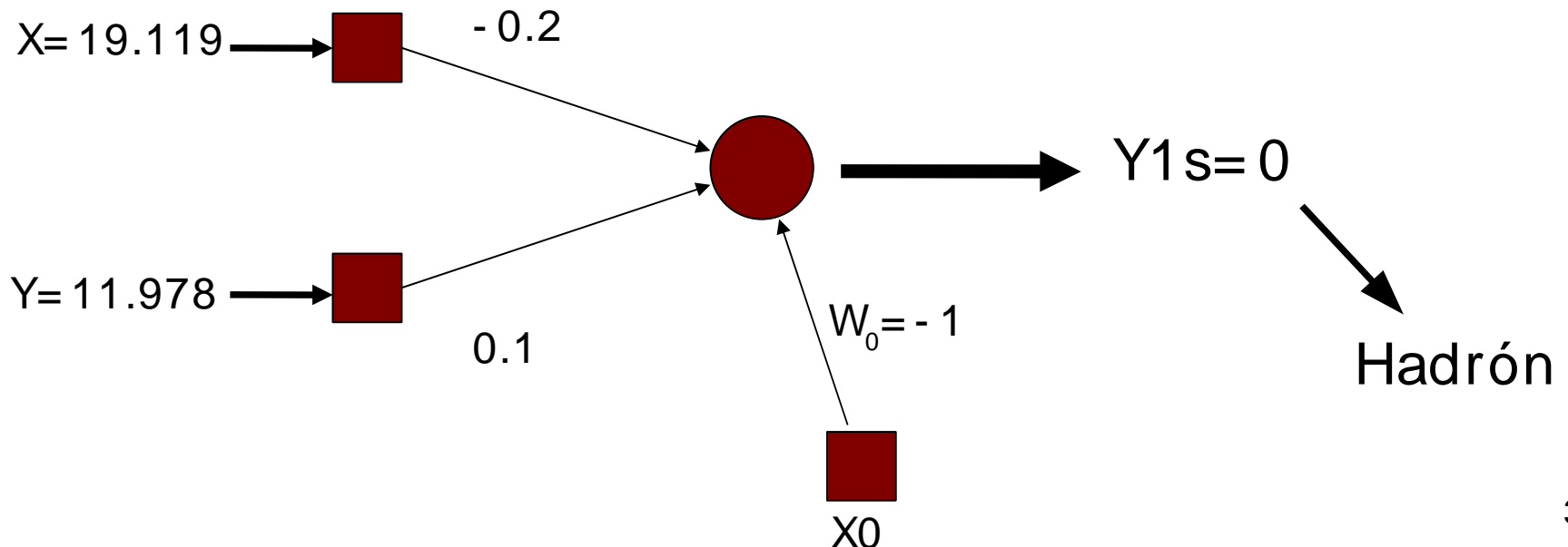
# El perceptrón simple: algoritmo

- En nuestro caso:
  - $\text{Netinput}_i = -1 + (-0.2) * 19.119 + 0.1 * 11.978 = -3.633$
  - $a_i = y_i = 0$



# El perceptrón simple: algoritmo

- En nuestro caso:
  - $\text{Netinput}_i = -1 + (-0.2) * 19.119 + 0.1 * 11.978 = -3.633$
  - $a_i = y_i = 0$



# El perceptrón simple: algoritmo

---

- Pero no es un hadrón, así que cambio mis pesos:

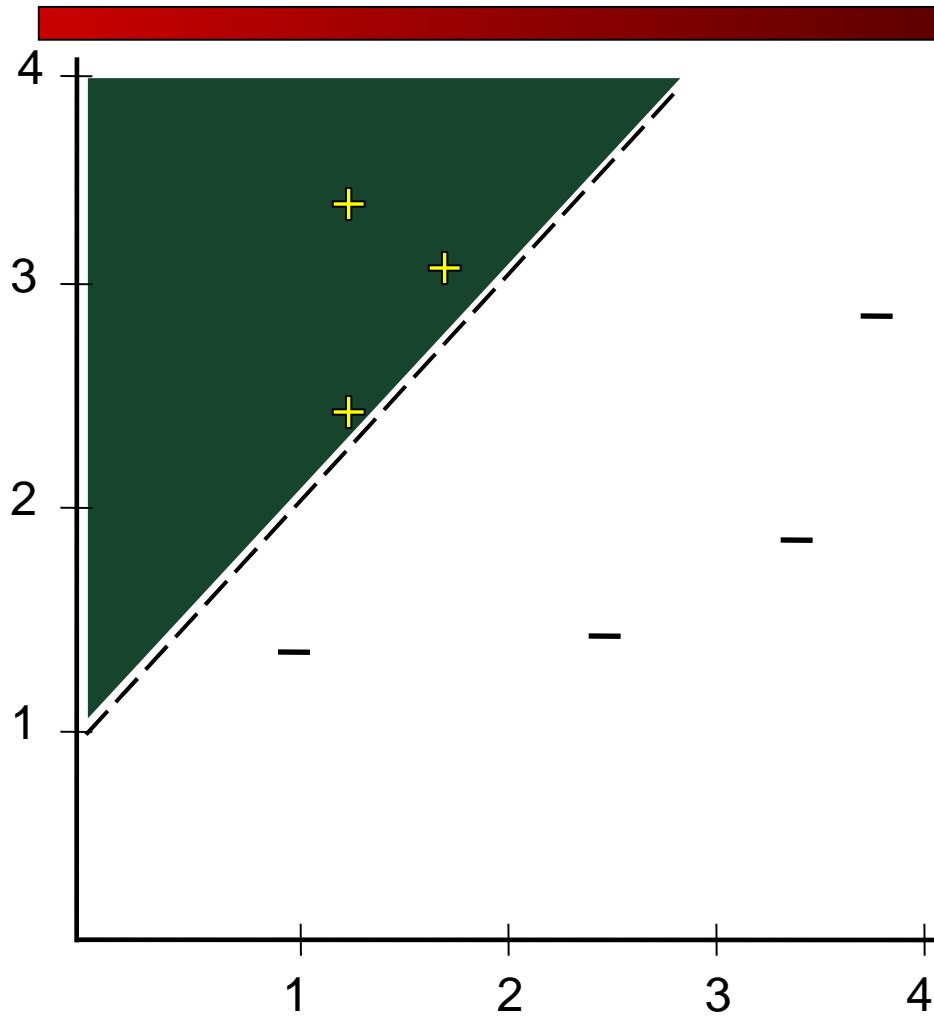
$$\Delta w_{11} = \epsilon x_1 \delta_1 = 0.4 * 19,119 * (1-0) = 7.64$$

$$\Delta w_{12} = \epsilon x_j \delta_i = 0.4 * 11,978 * (1-0) = 4,7912$$

- La próxima vez mi red lo hará mejor



# Regiones de decisión



$$1 + (-1)X + 1Y$$

$$w_0 = 1$$

$$w_{11} = -1$$

$$w_{12} = 1$$

Representa  
hiperplanos no  
necesariamente  
paralelos a los ejes.

**Conceptos  
linealmente  
separables**

# El perceptrón multicapa

---

- El perceptrón es el tipo más sencillo posible de red neuronal.
- Pero, ¿y si no son linealmente separables? – (Papert & Minsky, 1966)
- El problema famoso de la función XOR

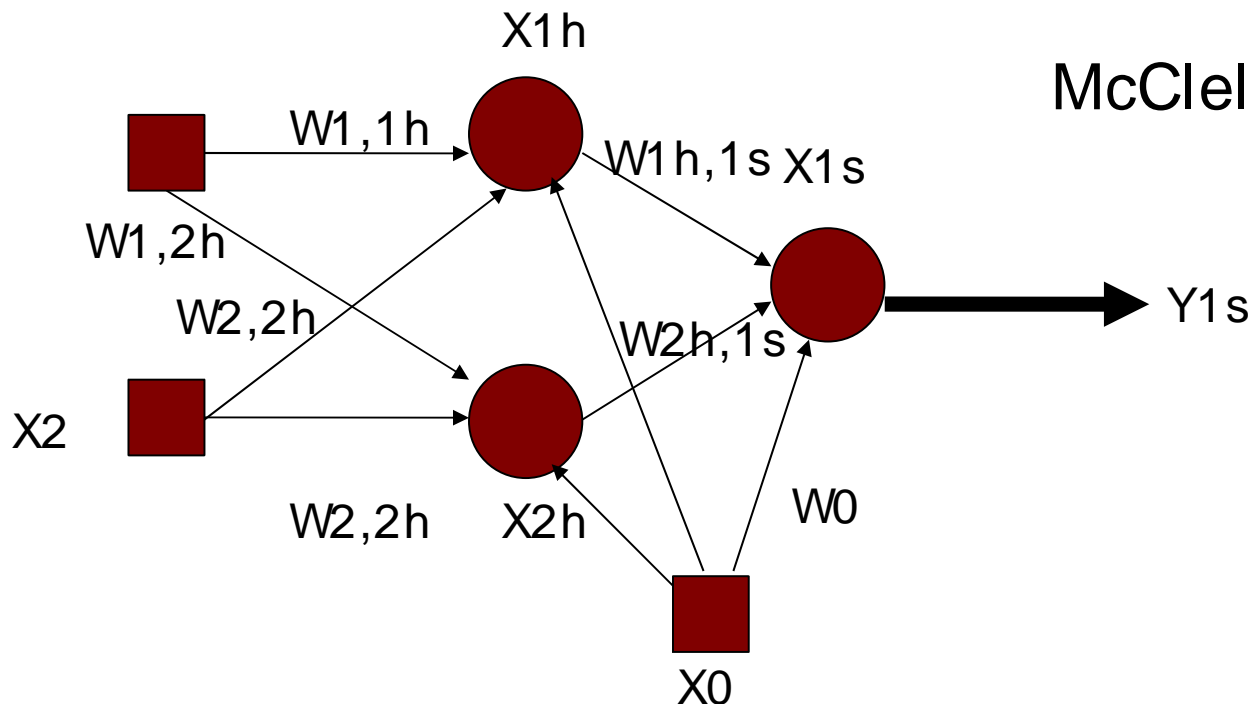
# El percetrón multicapa

---

- Podemos crear redes más complejas añadiendo más capas (**multilayer perceptron, MLP**).
- En la capa intermedia la representación de los patrones es linealmente separable.
- Y cambiar la regla de aprendizaje. **backproagation** (Rumelhart et al., 1986).

# El perceptrón multicapa

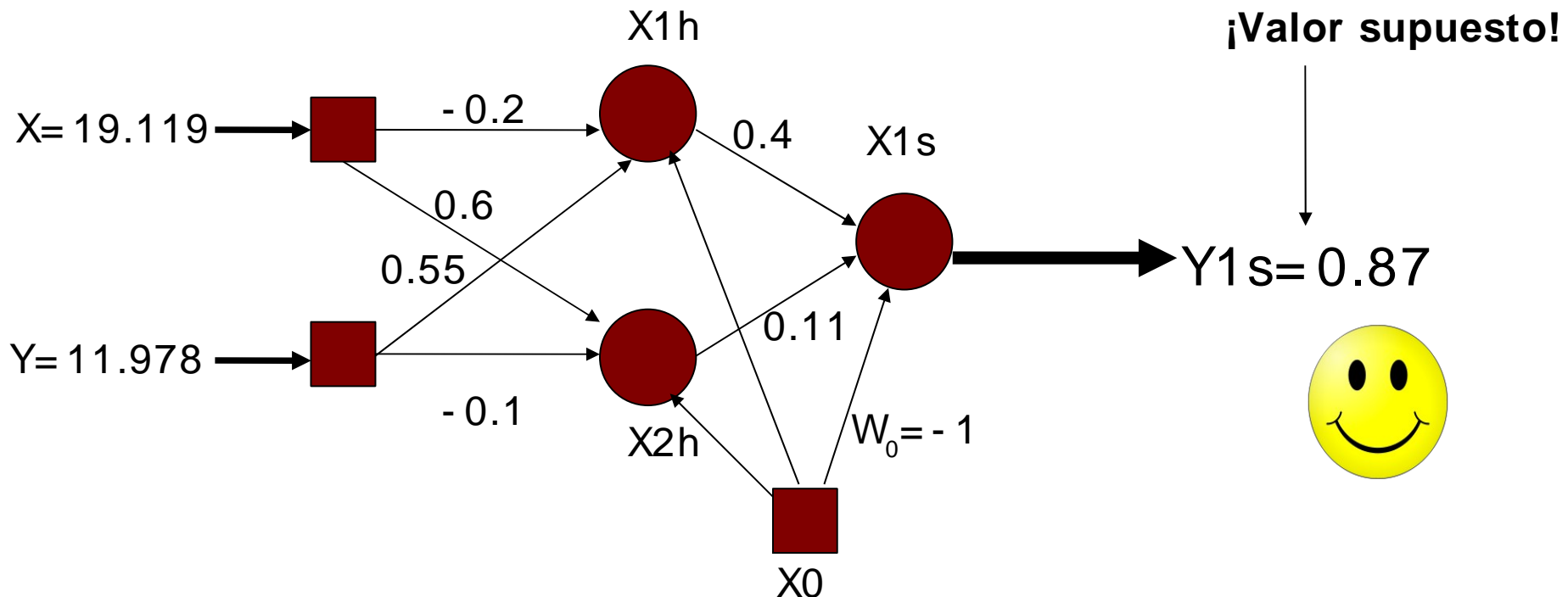
- Introducir la capa oculta (hidden)



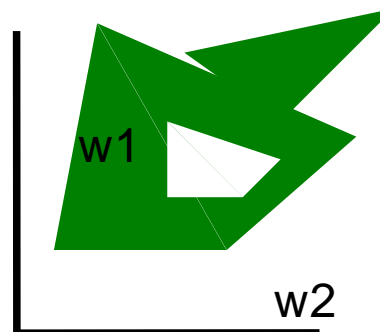
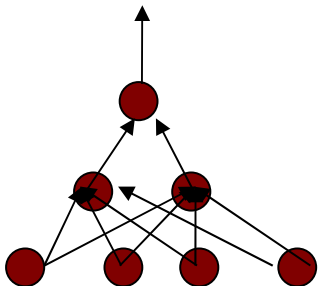
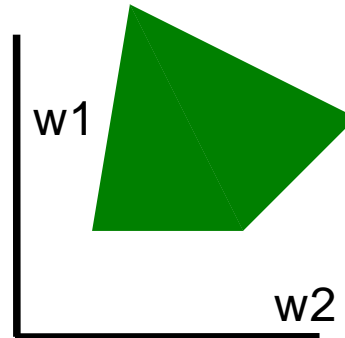
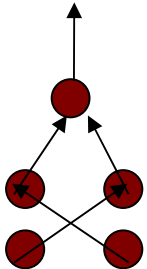
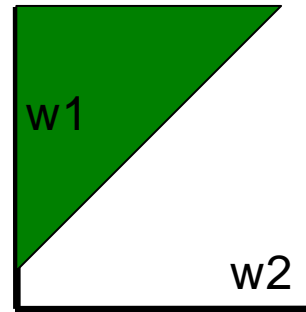
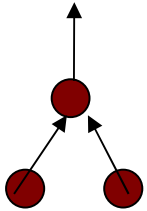
McClelland, 1986

# El perceptrón multicapa

- En nuestro caso



# Regiones de decisión MLP

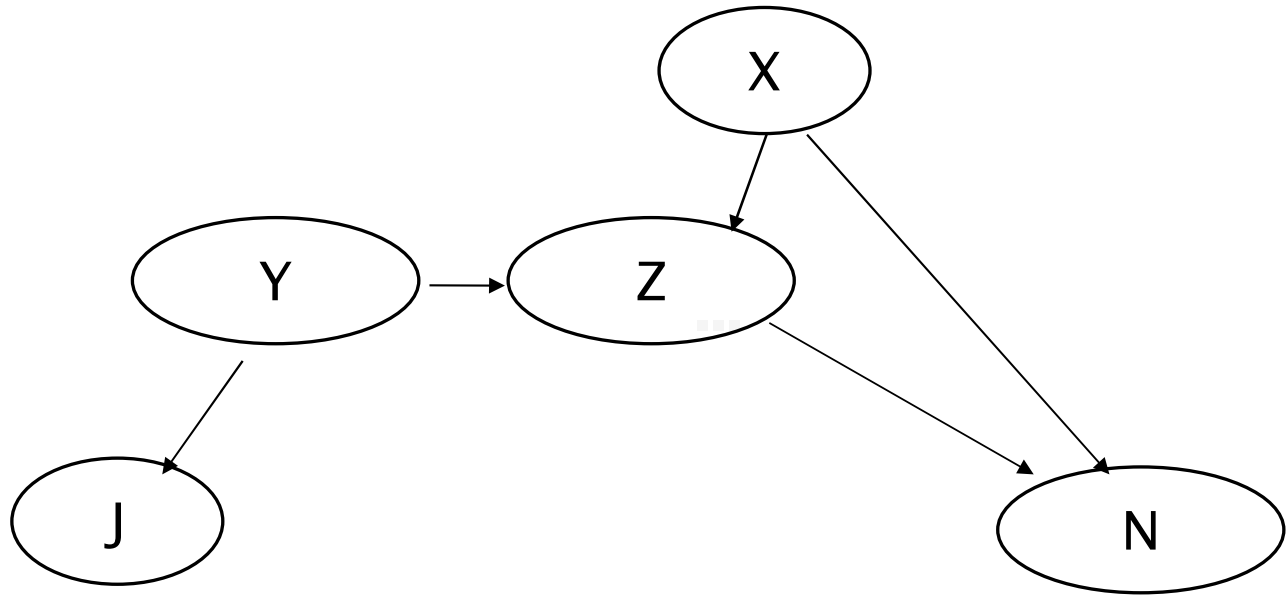


# Redes Bayesianas

---

# ¿Qué es una red bayesiana?

- Es un grafo dirigido acíclico (GDA)





# Redes Bayesianas

---

Cada nodo aleatorio viene descrito por:

$P(X=x_i)$	Nodos sin padres
$P(Z=z_k \mid X=x_i)$	probabilidad de observar el valor $z_i$ del atributo $Z$ conocido que el padre toma el valor $x_i$

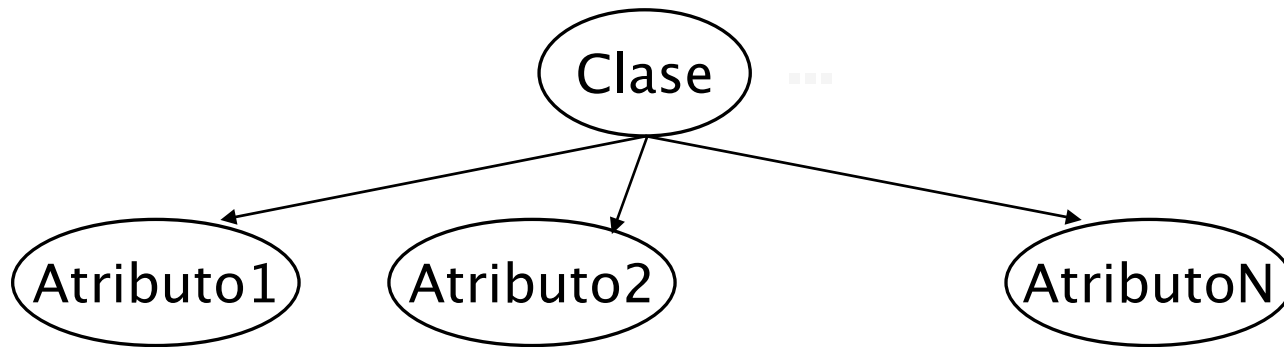
# Redes Bayesianas

---

- En principio relación causal
- Inferencia basada en:
  - Conocimiento de evidencias
  - Propiedades de independencia condicional (propiedades de Markov)
  - Teorema de Bayes

# ¿Qué es Naive Bayes?

- ¿Una red bayesiana ingenua?



# Redes Bayesianas

---

Un concepto se representa por:

$P(C_k)$	probabilidad de observar la clase k
$P(A_i=V_{ij} \mid C_k)$	probabilidad de observar el valor $V_{ij}$ del atributo $A_i$ en la clase k

# Redes Bayesianas

---

- No se construye un modelo
- **Atributos nominales:** la probabilidad se obtiene a partir de **frecuencias**.
- Se calcula el número de veces que se da cada valor para cada clase.

# Estimación de parámetros

---

- Por las propiedades de Markov, es suficiente con estimar la probabilidad a priori de la clase junto con las probabilidades condicionadas de cada atributo dada la clase.
- MAP (maximum a posteriori)

$$\text{classify}(f_1, \dots, f_n) = \operatorname{argmax}_c P(C = c) \prod_{i=1}^n p(F_i = f_i | C = c)$$

# Ejemplo de aplicación

- En nuestro ejemplo (discretizando)

Eje mayor elipse (X)	Eje menor elipse (Y)	Gamma
Corto	Corto	Sí
Largo	Corto	Sí
Largo	Medio	Sí
Medio	Medio	Sí
Medio	Corto	No
Corto	Medio	No
Largo	Largo	No

# Estimación de parámetros

---

- En nuestro ejemplo: Si viene un ejemplo ( $X=\text{corto}$ ,  $Y=\text{medio}$ )
  - $P(\text{Gamma})=4/7$
  - $P(X=\text{corto}|\text{gamma})= 1/4$
  - $P(Y=\text{medio}|\text{gamma})= 2/4$
  - $p(\text{gamma}|X=\text{corto}, Y=\text{medio})=4/7*1/4*2*4=1/14=0.07$
  - \*\*\*\*\*
  - $P(\text{Hadrón})=3/7$
  - $P(X=\text{corto}|\text{hadrón})= 1/3$
  - $P(Y=\text{medio}|\text{hadrón})= 1/3$
  - $p(\text{hadrón}|X=\text{corto}, Y=\text{medio})=3/7*1/3*2*3=2/21=0.095$



# Clasificación

---

- Así que lo clasificamos como hadrón.
- Con probabilidad  $0.095 / 0.07 + 0.095 = 0.5757$

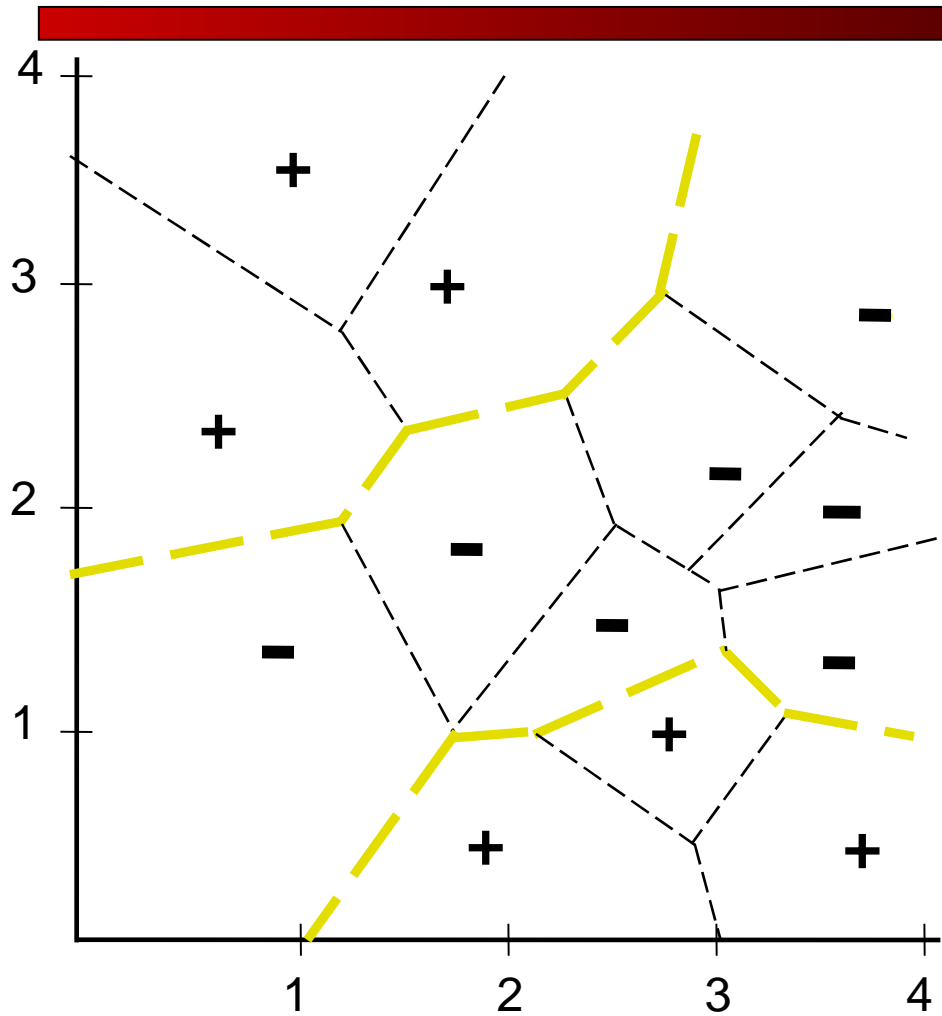
¿Podemos concluir que es un hadrón?

# Aprendizaje basado en ejemplos

---

- Tampoco se construye un modelo, simplemente almacenamos las instancias (Lazy Learning).
- Todo el trabajo se realiza en el momento de la predicción.
- Basado en la noción de prototipo.
- Para realizar una predicción se busca la instancia más similar.

# Regiones de decisión



Cada instancia genera una región de decisión **convexa** (diagramas de Voronoi)

Pero la combinación no tiene porqué serlo.

# K vecinos más cercanos

---

- **Algoritmo K-NN.** Calcular la distancia utilizando las  $k$  instancias más cercanas
- La clase se elige por votación.
- **Modificación:** Dar diferentes pesos en la votación en función de la distancia al ejemplo que se quiere clasificar.

# Aprendizaje basado en instancias

---

- Problemas:
  - Es costoso (computación)
  - No tolera muy bien el ruido (ampliar K)
  - Atributos discretos
  - Valores desconocidos para un atributo

# Ejemplo de aplicación

## ■ Nuestro ejemplo

Eje mayor elipse (X)	Eje menor elipse (Y)	Gamma	Distancias
19,119	11,978	Sí	36,45
88,709	8,637	Sí	88,74
63,100	21,900	Sí	64,06
48,050	13,550	Sí	48,13
69,989	18,122	No	70,37
22,777	12,708	No	22,85
121,603	59,445	No	130,96

55,55

10,84

# Ejemplo de aplicación

## ■ Nuestro ejemplo

Eje mayor elipse (X)	Eje menor elipse (Y)	Gamma	Distancias
19,119	11,978	Sí	36,45
88,709	8,637	Sí	88,74
63,100	21,900	Sí	64,06
48,050	13,550	Sí	48,13
69,989	18,122	No	70,37
22,777	12,708	No	22,85
121,603	59,445	No	130,96

55,55

10,84

K=1  
HADRÓN

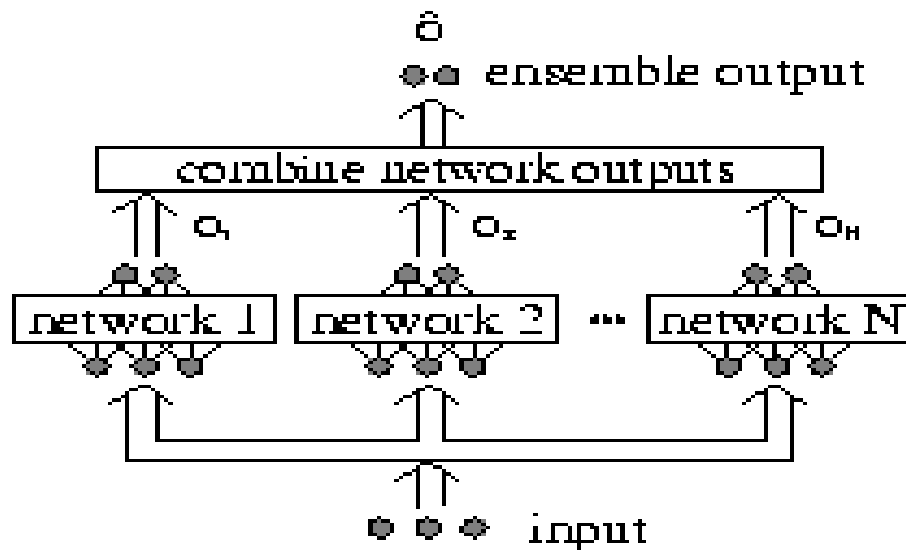
# Combinación de clasificadores

---



# Combinación de clasificadores

- Hablaremos de metaclassificadores



Opitz, (1999)

# Combinación de clasificadores

---

- Comité de expertos
- En la predicción la mayoría gana
- Se necesita diversidad en el error cometido por los modelos

# Bagging

---

- *Bootstrap aggregating*
- Dado un conjunto de entrenamiento  $D$  de tamaño  $N$ , se generan  $K$  nuevos conjuntos de entrenamiento de tamaño  $N'$  ( $N' < N$ )
- Se muestrean ejemplos de  $D$ , y pueden aparecer repetidos en los conjuntos obtenidos.

# Bagging

---

- Generamos entonces un modelo para cada nuevo conjunto de entrenamiento
- Combinación:
  - Por votación (clasificación)
  - Sacando la media de las predicción (regresión)
- Reduce el error (reduciendo la varianza)
- Típico usarlo con árboles de decisión (pero no tiene por qué ser así)

# Otras combinaciones

---

- *Boosting*
  - *Adaboost (Freund, 1996)* – la salida de un modelo influye en el siguiente, poniendo énfasis en los errores del modelo anterior
- *Random Forest* – variante de bagging
- Combinadores híbridos (o heterogéneos)
  - Se pueden combinar modelos de diferentes tipos
  - Pueden haber grados (clasificadores base y árbitro)

# Fase 3: Evaluación de los modelos

---

# Evaluación de los modelos

---

- Esto lo veremos después



# Referencias

---

**Aha D. W., Kibler D. & Albert M. K. (1991).** Instance based learning algorithms. *Machine Learning* 6, 37–66.

**Bishop C. (1995).** *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford.

**Breiman L., Friedman J., Olshen R. & Stone C.(1984).** *Classification and Regression Trees*, Wadsworth International

**Breiman L. (1996).** Bagging Predictors, *Machine Learning*, Volume 24, Number 2 / August, pages 123–140

**Domingos P.& Pazzani M. (1997).** On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, *Machine Learning*, 29(2–3), 103–130

**Freund Y., Schapire R. E., (1996).** Experiments with a new boosting algorithm, *Machine Learning: Proceedings of the Thirteenth International Conference (ICML '96)*

**Kuncheva L.(2004)** *Combining pattern classifiers*. Wiley.



# Referencias

---

**Langley P., Iba W. & Thompson K. (1992). An analysis of Bayesian classifiers. *Proceedings of the Tenth National Conference on Artificial Intelligence*, 223–228.**

**Lippmann R. P. (1987). An introduction to computing with neural nets, *IEEE ASSP Magazine*, 4–22.**

**McClelland J.L. , Rumelhart D.E. and the PDP Research Group (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models, Cambridge, MA: MIT Press**

**Minsky M. ,Papert S. (1969) Perceptrons (Cambridge, MA: MIT Press)**

**Mitchell T. M. (1997). *Machine Learning*, McGraw Hill.**

**Opitz,D., Maclin R., (1999). Popular ensemble methods: An empirical study, *Journal of Artificial Intelligence Research*, Volume 11, pages 169–198**

**Quinlan J. R. (1986). Induction of Decision trees, *Machine Learning* 1(1), 81–106.**

**Quinlan J. R. (1992). Learning with Continuous Classes. *Proceedings of AI92*.**

# Referencias

---

**Rosenblatt F. (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408.**

**Rumelhart D. E., Hinton G., Williams R. J. (1986).** Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the microstructure of cognition (Vol. 1)*. MIT Press.

**Ting, K. M., & Witten, I. H..** Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10 (1999) 271–289

**Wolpert, D.** Stacked generalization. *Neural Networks* 5(2). (1992) 241–260

# ¡Muchas gracias!

---

- Turno de agresiones y preguntas (mejor primero las preguntas)