Tema 6: Introducción al aprendizaje automático

M. A. Gutiérrez Naranjo
F. J. Martín Mateos
J. L. Ruiz Reina

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Contenido

- Introducción
- Aprendizaje de árboles de decisión
- Aprendizaje de reglas
- Aprendizaje basado en instancias: kNN
- *Clustering*

Sección 1

Sección 1

Introducción

¿Qué es aprender?

Definiciones de *aprendizaje*:

- Cualquier *cambio* en un sistema que le permita realizar la misma tarea de manera *más eficiente* la próxima vez (*H. Simon*)
- *Modificar la representación* del mundo que se está percibiendo (*R. Michalski*)
- Realizar cambios *útiles* en nuestras mentes (*M. Minsky*)
- Se dice que aprendemos de la *experiencia* a realizar alguna *tarea* si la realización de la tarea mejora con la experiencia respecto a alguna *medida de rendimiento* (*T. M. Mitchell*)

¿Qué es aprender?

Definiciones de *aprendizaje*:

- Cualquier *cambio* en un sistema que le permita realizar la misma tarea de manera *más eficiente* la próxima vez (*H. Simon*)
- *Modificar la representación* del mundo que se está percibiendo (*R. Michalski*)
- Realizar cambios *útiles* en nuestras mentes (*M. Minsky*)
- Se dice que aprendemos de la *experiencia* a realizar alguna *tarea* si la realización de la tarea mejora con la experiencia respecto a alguna *medida de rendimiento* (*T. M. Mitchell*)

¿Qué es aprender?

Definiciones de *aprendizaje*:

- Cualquier *cambio* en un sistema que le permita realizar la misma tarea de manera *más eficiente* la próxima vez (*H. Simon*)
- *Modificar la representación* del mundo que se está percibiendo (*R. Michalski*)
- Realizar cambios *útiles* en nuestras mentes (*M. Minsky*)
- Se dice que aprendemos de la *experiencia* a realizar alguna *tarea* si la realización de la tarea mejora con la experiencia respecto a alguna *medida de rendimiento* (*T. M. Mitchell*)

¿Qué es aprender?

Definiciones de *aprendizaje*:

- Cualquier *cambio* en un sistema que le permita realizar la misma tarea de manera *más eficiente* la próxima vez (*H. Simon*)
- *Modificar la representación* del mundo que se está percibiendo (*R. Michalski*)
- Realizar cambios *útiles* en nuestras mentes (*M. Minsky*)
- Se dice que aprendemos de la *experiencia* a realizar alguna *tarea* si la realización de la tarea mejora con la experiencia respecto a alguna *medida de rendimiento* (*T. M. Mitchell*)

Aprendizaje

- Aprendizaje automático: construir sistemas computacionales que a partir de datos y percepciones mejoran su rendimiento en la realización de una determinada tarea (sin haber sido explicitamente programados para dicha tarea).
- Ejemplos de tareas:
  - Construcción de bases de conocimiento a partir de experiencia y observaciones
  - Clasificación y diagnóstico
  - Minería de datos, descubrir modelos matemáticos que explican un fenómeno a partir de gran cantidad de datos
  - Resolución de problemas, planificación y acción

Tipos de aprendizaje y paradigmas

- Tipos de aprendizaje
  - Supervisado
  - No supervisado
  - Con refuerzo
- Paradigmas
  - Aprendizaje basado en instancias (memorización)
  - Agrupamiento (*Clustering*)
  - Aprendizaje inductivo
  - Aprendizaje por analogía
  - Descubrimiento
  - Algoritmos genéticos, redes neuronales

Ejemplo de aprendizaje

- Conjunto de entrenamiento
  - Ejemplos: días en los que es recomendable (o no) jugar al tenis
  - Representación como una lista de pares atributo–valor

Ej.	Cielo	Temperatura	Humedad	Viento	JugarTenis
D_1	Soleado	Alta	Alta	Débil	-
D_2	Soleado	Alta	Alta	Fuerte	-
D_3	Nublado	Alta	Alta	Débil	+
D_4	Lluvia	Suave	Alta	Débil	+
...					

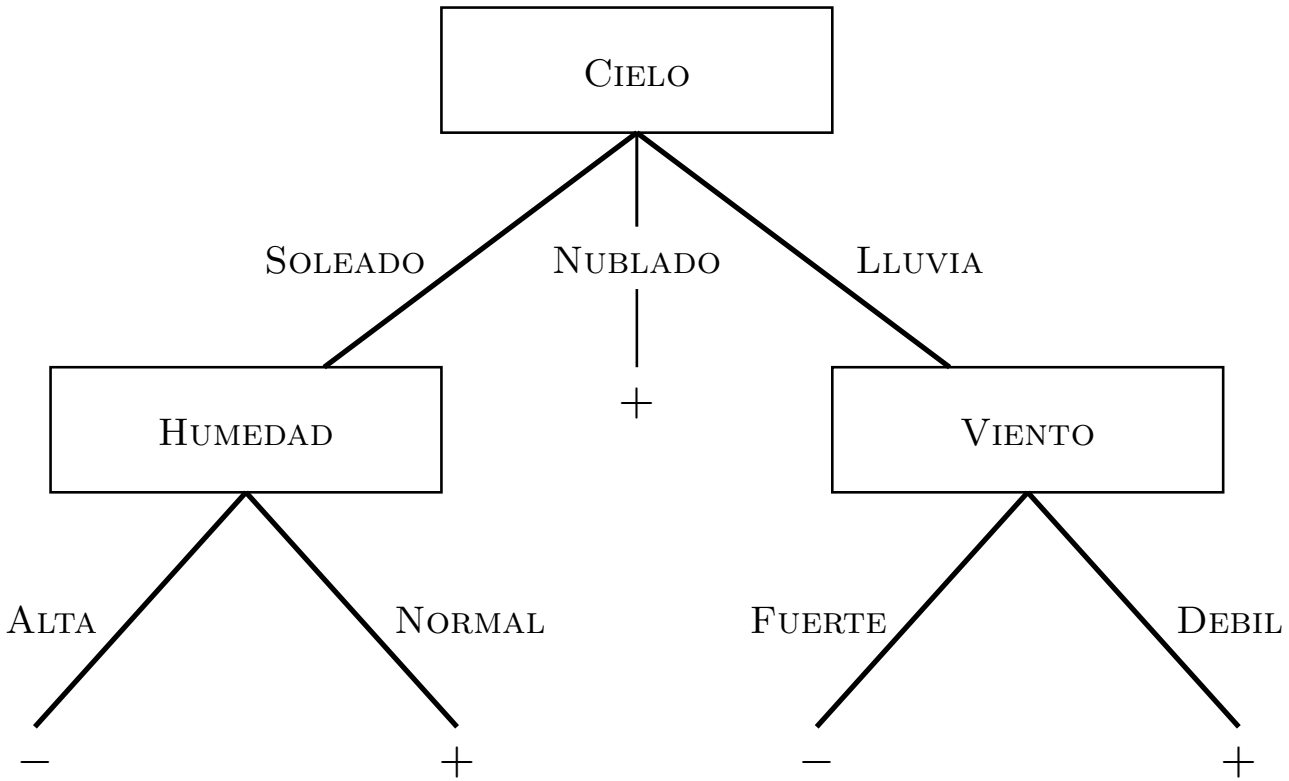
- Objetivo: Dado el *conjunto de entrenamiento*, *aprender* el concepto "Días en los que se juega al tenis"
  - Se trata de *aprendizaje supervisado*
- Problema: ¿Cómo expresar lo aprendido?
  - En este tema, veremos algoritmos para aprender *árboles de decisión*, *reglas*, *modelos probabilísticos*,...

Sección 2

Sección 2
Aprendizaje de árboles de decisión

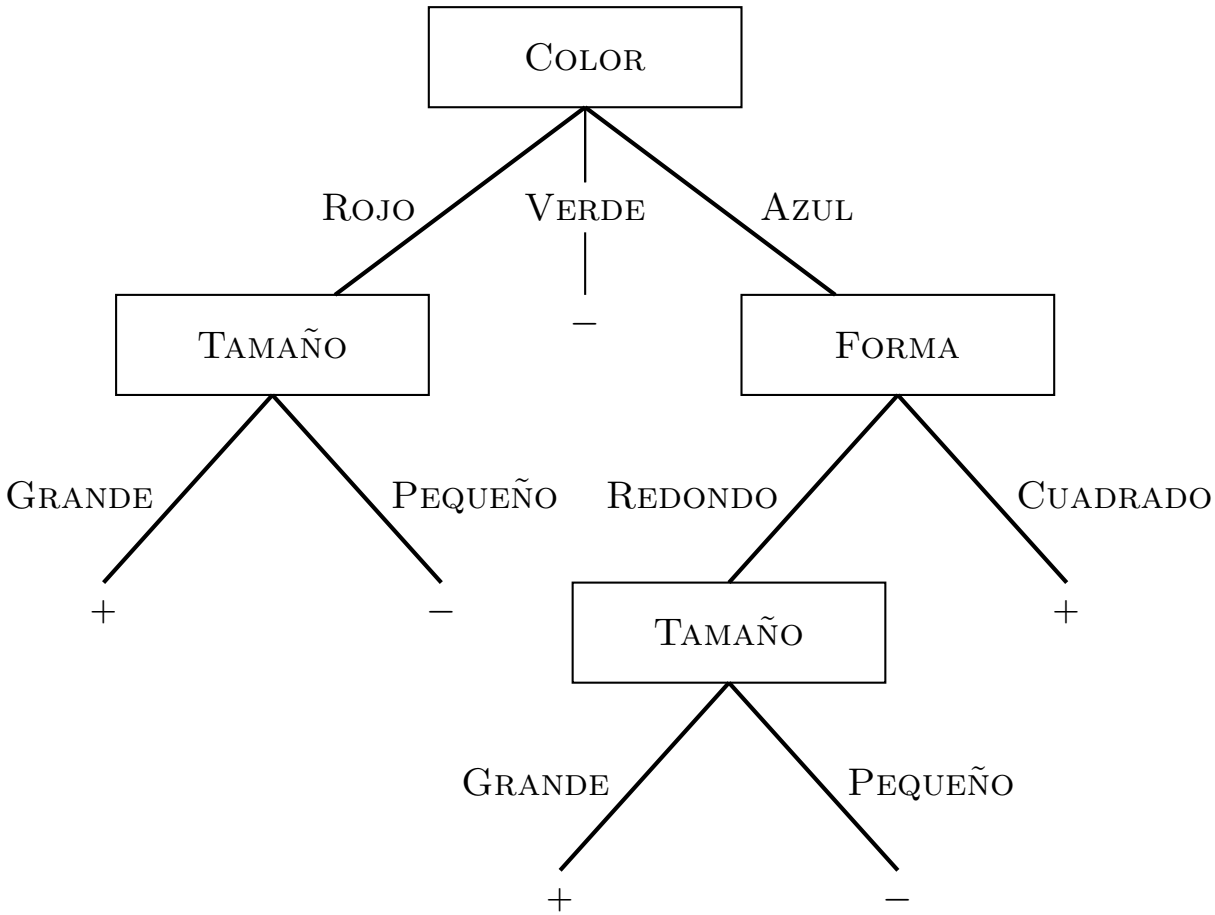
Árboles de decisión

- Un árbol de decisión es un grafo etiquetado que representa un concepto.
- Ejemplos de árboles de decisión



Árboles de decisión

- Ejemplos de árboles de decisión



Árboles de decisión

- Árboles de decisión
  - Nodos interiores: atributos
  - Arcos: posibles valores del nodo origen
  - Hojas: valor de clasificación (usualmente $+$ ó $-$, aunque podría ser cualquier conjunto de valores, no necesariamente binario)
  - Representación de una función objetivo
- Disyunción de reglas proposicionales:

$$
\begin{aligned}
&(\textsc{Cielo}{=}\textsc{Soleado} \wedge \textsc{Humedad}{=}\textsc{Alta} \rightarrow \textsc{JugarTenis}{=}-)\\
\vee\ &(\textsc{Cielo}{=}\textsc{Soleado} \wedge \textsc{Humedad}{=}\textsc{Normal} \rightarrow \textsc{JugarTenis}{=}+)\\
\vee\ &(\textsc{Cielo}{=}\textsc{Nublado} \rightarrow \textsc{JugarTenis}{=}+)\\
\vee\ &(\textsc{Cielo}{=}\textsc{Lluvioso} \wedge \textsc{Viento}{=}\textsc{Fuerte} \rightarrow \textsc{JugarTenis}{=}-)\\
\vee\ &(\textsc{Cielo}{=}\textsc{Lluvioso} \wedge \textsc{Viento}{=}\textsc{Debil} \rightarrow \textsc{JugarTenis}{=}+)
\end{aligned}
$$

- Capaz de representar cualquier subconjunto de instancias

Aprendizaje de árboles de decisión

- Objetivo: aprender un árbol de decisión consistente con los ejemplos, para posteriormente clasificar ejemplos nuevos
- Ejemplo de conjunto de entrenamiento:

Ej.	Cielo	Temperatura	Humedad	Viento	JugarTenis
D_1	Soleado	Alta	Alta	Débil	-
D_2	Soleado	Alta	Alta	Fuerte	-
D_3	Nublado	Alta	Alta	Débil	+
D_4	Lluvia	Suave	Alta	Débil	+
...					

Árboles de decisión

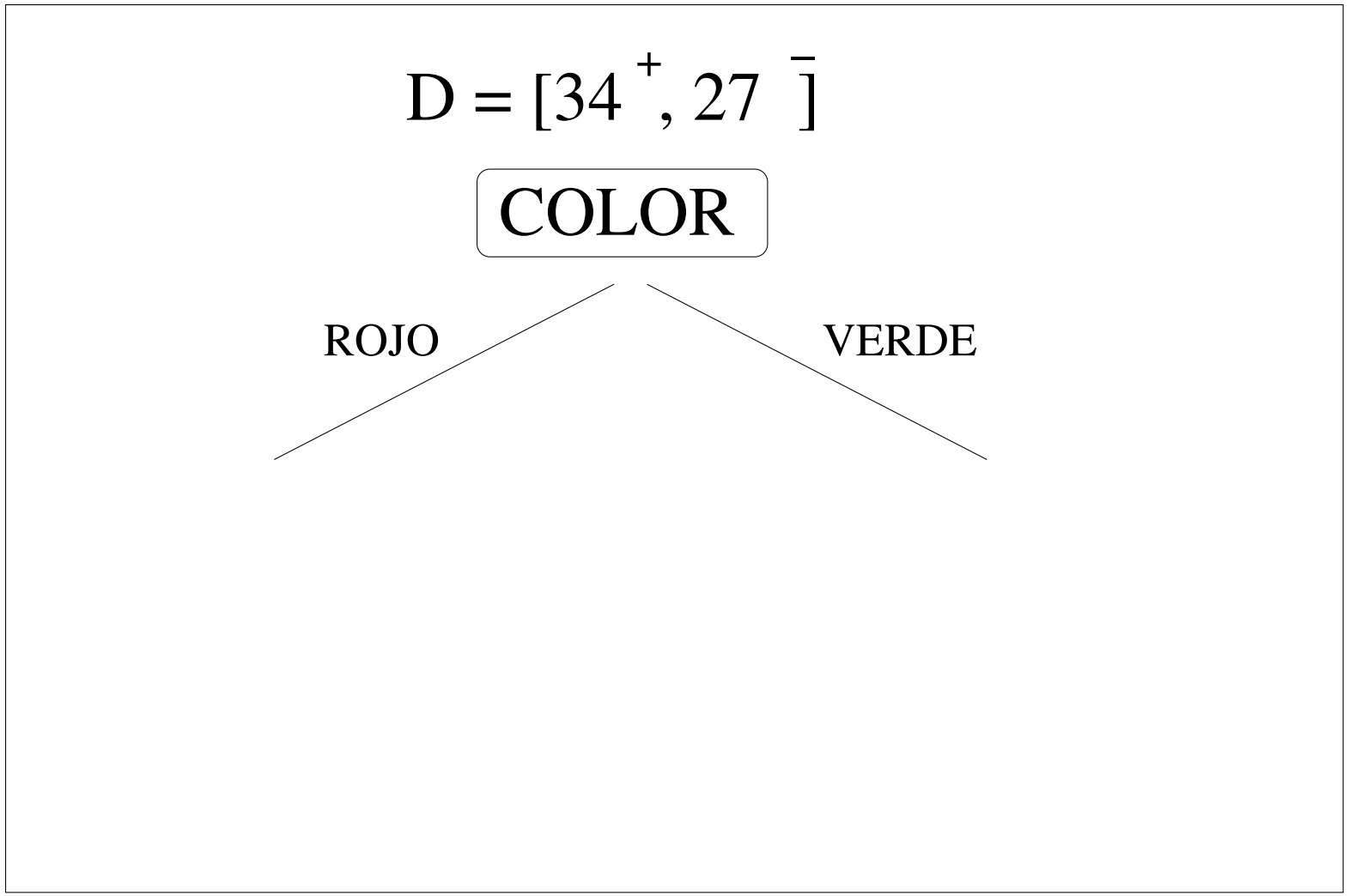
$$D = [34^{+}, 27^{-}]$$

Árboles de decisión

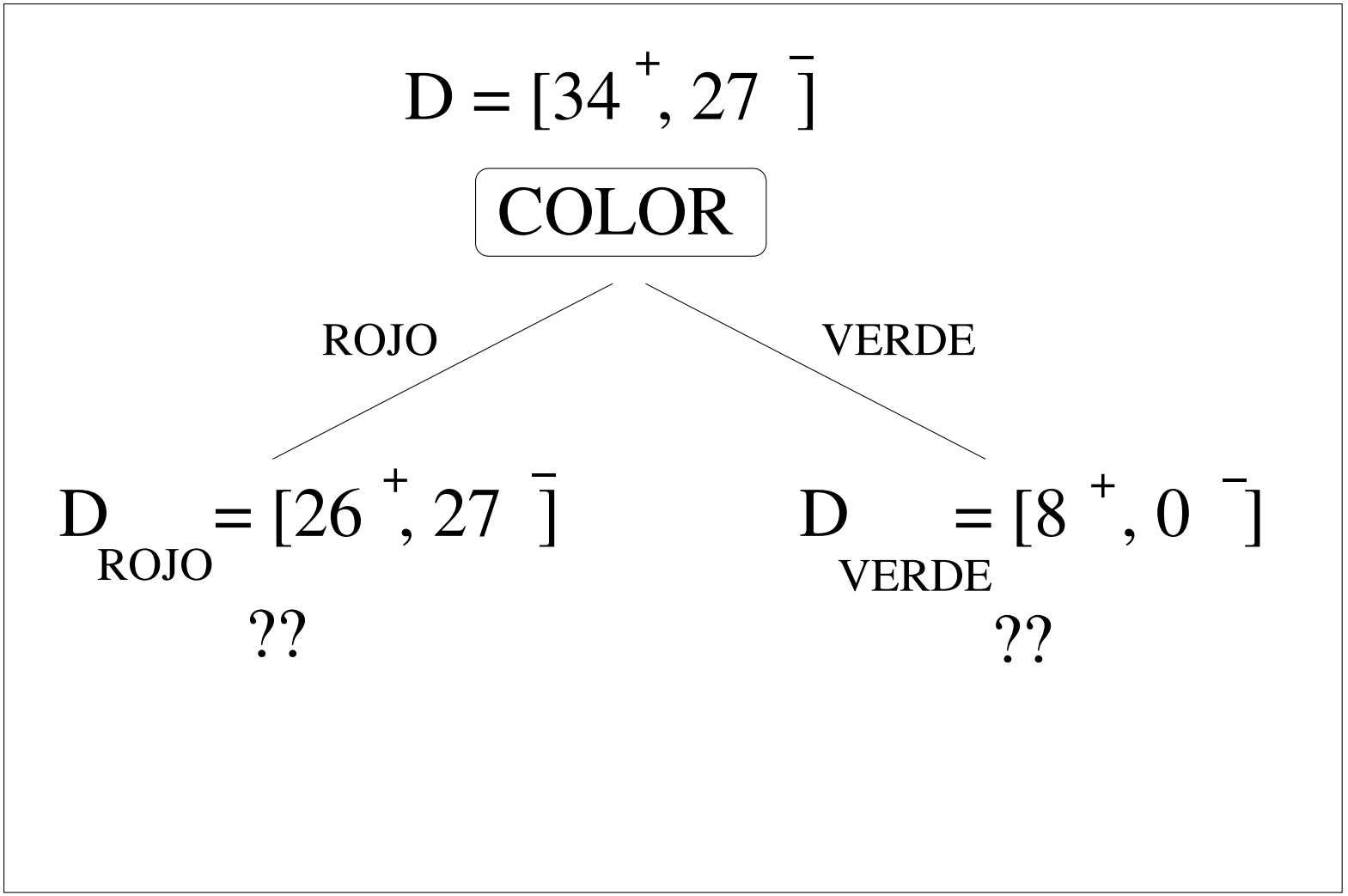
$$D = [34^{+}, 27^{-}]$$

COLOR

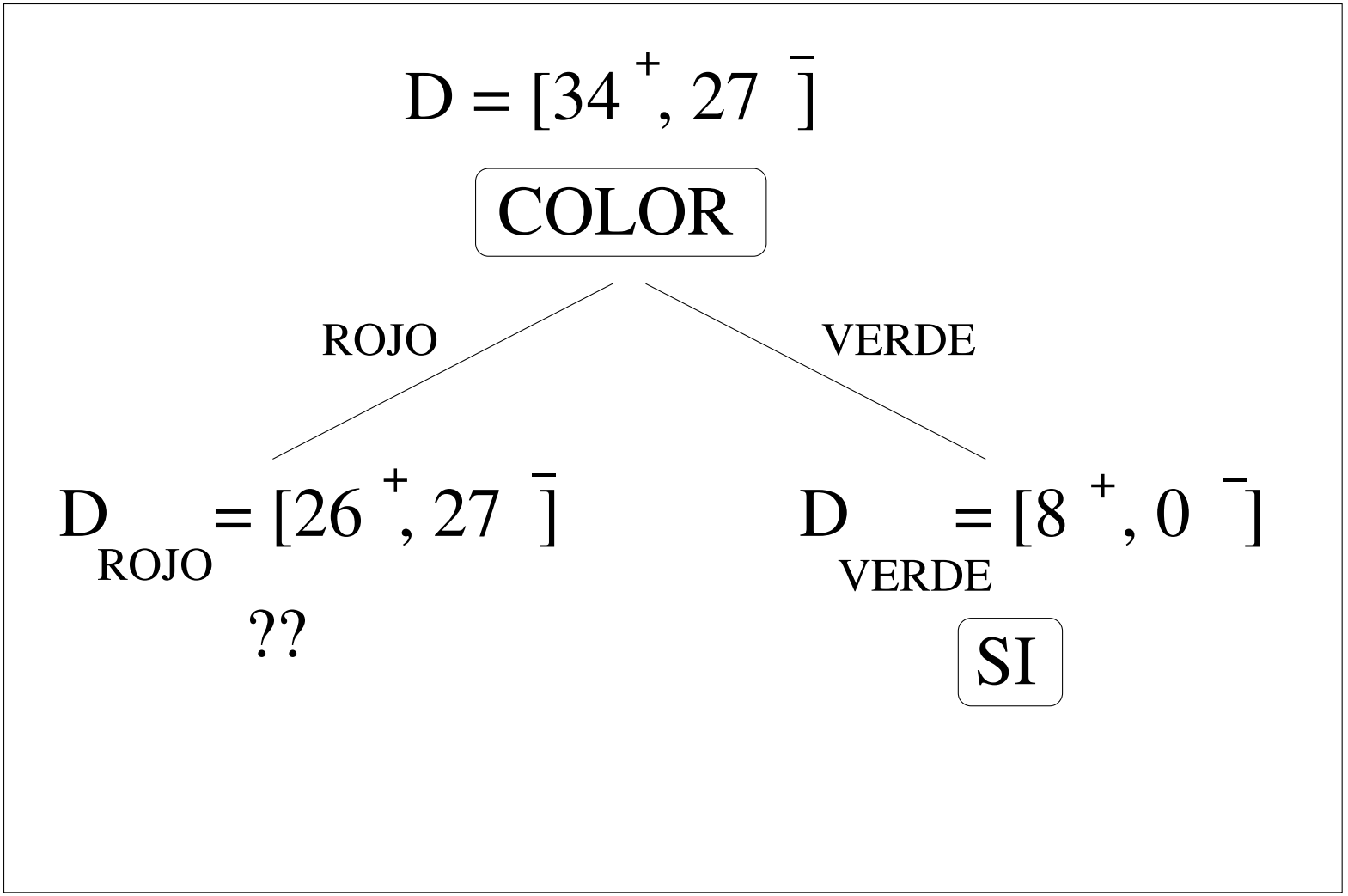
Árboles de decisión



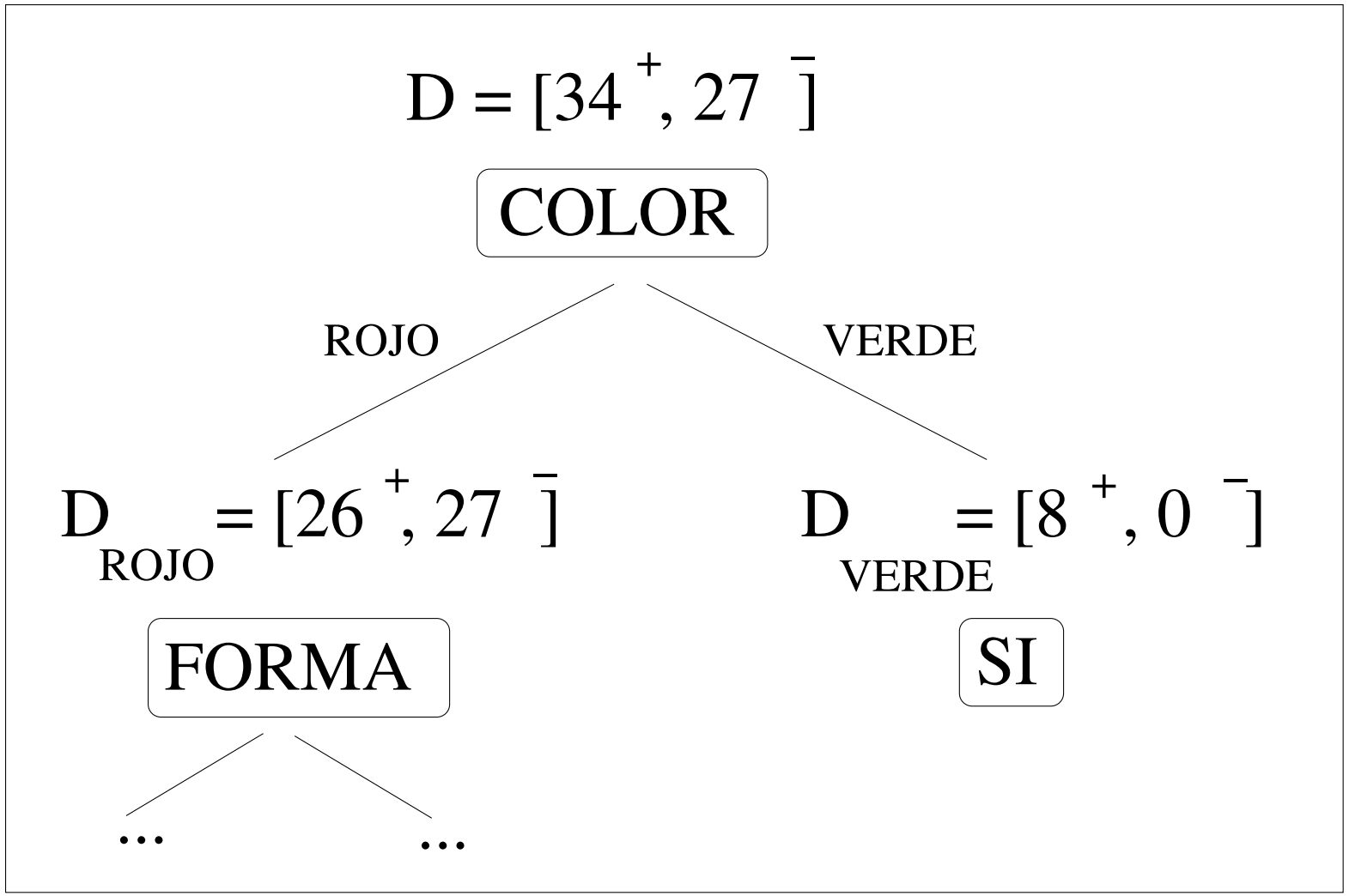
Árboles de decisión



Árboles de decisión

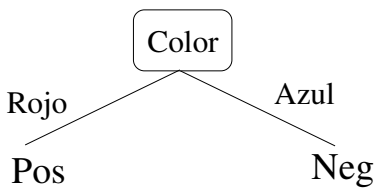


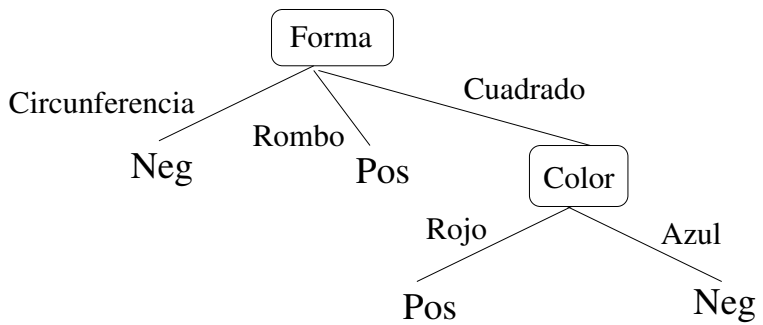
Árboles de decisión



Clasificadores

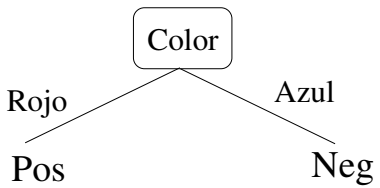
Color	Forma	Clasificación
Rojo	Cuadrado	Pos
Rojo	Rombo	Pos
Azul	Circunferencia	Neg
Azul	Cuadrado	Neg

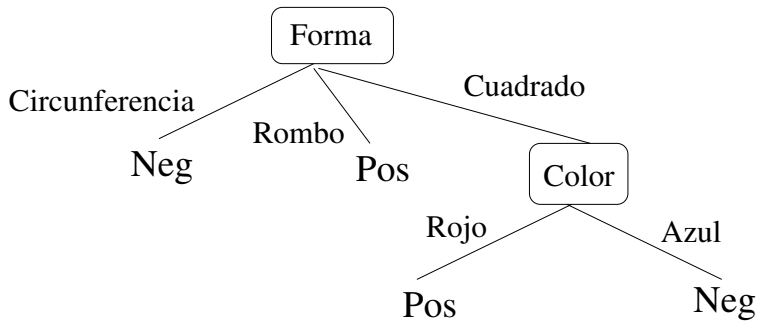




Clasificadores

Color	Forma	Clasificación
Rojo	Cuadrado	Pos
Rojo	Rombo	Pos
Azul	Circunferencia	Neg
Azul	Cuadrado	Neg





$\langle Azul, Rombo \rangle$???

La navaja de Occam

Guillermo de Occam (1288-1349)

Lex parsimoniae

Entia non sunt multiplicanda praeter necessitatem (No ha de presumirse la existencia de más cosas que las absolutamente necesarias)

Guillermo de Occam

La navaja de Occam

En igualdad de condiciones la solución más sencilla es probablemente la correcta

Algoritmo ID3

Algoritmo ID3

```
ID3(Ejemplos, Atributo-objetivo, Atributos)
1. Si todos los Ejemplos son positivos, devolver un nodo
   etiquetado con +
2. Si todos los Ejemplos son negativos, devolver un nodo
   etiquetado con -
3. Si Atributos está vacío, devolver un nodo etiquetado con el valor
   más frecuente de Atributo-objetivo en Ejemplos.
4. En otro caso:
4.1. Sea A el atributo de Atributos que MEJOR clasifica Ejemplos
4.2. Crear Árbol, con un nodo etiquetado con A.
4.3. Para cada posible valor v de A, hacer:
   * Añadir un arco a Árbol, etiquetado con v.
   * Sea Ejemplos(v) el subconjunto de Ejemplos con valor del
     atributo A igual a v.
   * Si Ejemplos(v) es vacío:
     - Entonces colocar debajo del arco anterior un nodo etiquetado
       con el valor más frecuente de Atributo-objetivo en Ejemplos.
     - Si no, colocar debajo del arco anterior el subárbol
       ID3(Ejemplos(v), Atributo-objetivo, Atributos-{A}).
4.4  Devolver Árbol
```

¿Cómo saber qué atributo clasifica mejor?

- Entropía de un conjunto de ejemplos D (resp. de una clasificación):

$$Ent(D) = -\frac{|P|}{|D|} \cdot log_2 \frac{|P|}{|D|} - \frac{|N|}{|D|} \cdot log_2 \frac{|N|}{|D|}$$

  donde P y N son, respectivamente, los subconjuntos de ejemplos positivos y negativos de D
  - Notación: $Ent([p^+, n^-])$, donde $p = |P|$ y $n = |N|$
- Intuición:
  - Mide la ausencia de "homegeneidad" de la clasificación
  - Teoría de la Información: cantidad media de información (en bits) necesaria para codificar la clasificación de un ejemplo
- Ejemplos:
  - $Ent([9^+, 5^-]) = -\frac{9}{14} \cdot log_2 \frac{9}{14} - \frac{5}{14} \cdot log_2 \frac{5}{14} = 0{,}94$
  - $Ent([k^+, k^-]) = 1$ (ausencia total de homogeneidad)
  - $Ent([p^+, 0^-]) = Ent([0^+, n^-]) = 0$ (homogeneidad total)

Ganancia de información

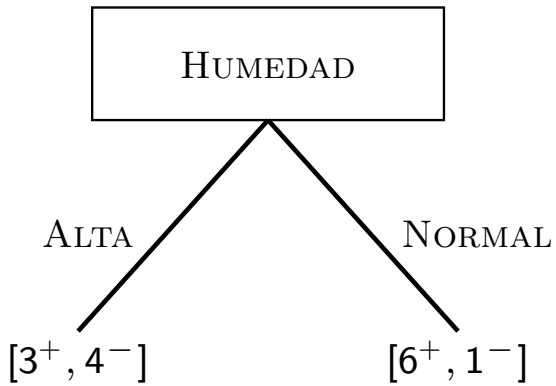
- Preferimos nodos con menos entropía (árboles pequeños)
- Entropía *esperada* después de usar un atributo A en el árbol:
  $$\sum_{v \in Valores(A)} \frac{|D_v|}{|D|} \cdot Ent(D_v)$$
  donde D_v es el subconjunto de ejemplos de D con valor del atributo A igual a v
- Ganancia de información *esperada* después de usar un atributo A:
  $$Ganancia(D, A) = Ent(D) - \sum_{v \in Valores(A)} \frac{|D_v|}{|D|} \cdot Ent(D_v)$$
- En el algoritmo ID3, en cada nodo usamos el atributo con mayor ganancia de información (considerando los ejemplos correspondientes al nodo)

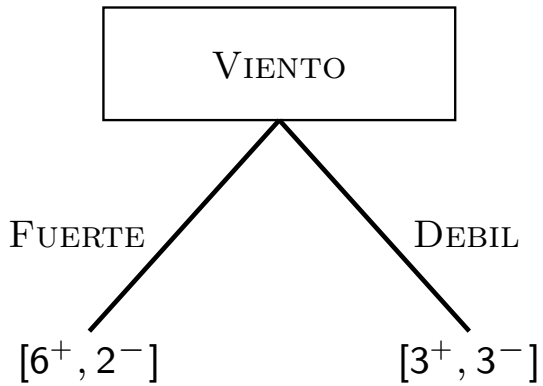
Algoritmo ID3 (ejemplo 1)

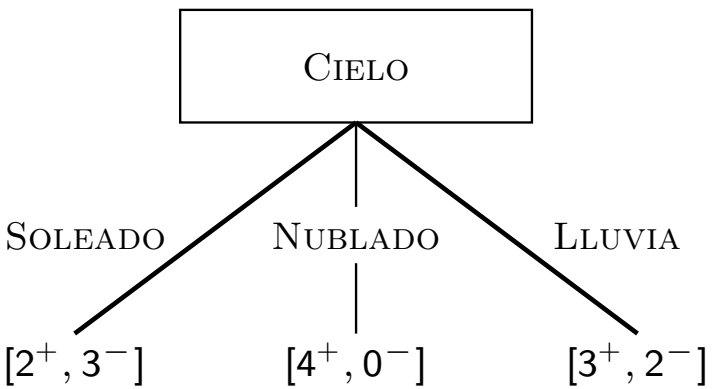
- Conjunto de entrenamiento:

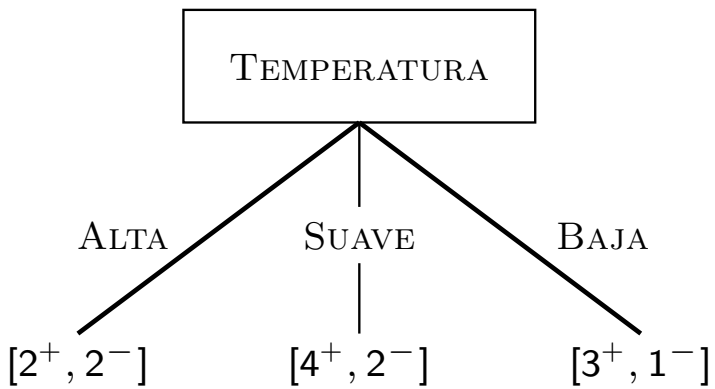
Ej.	Cielo	Temperatura	Humedad	Viento	JugarTenis
D_1	Soleado	Alta	Alta	Débil	-
D_2	Soleado	Alta	Alta	Fuerte	-
D_3	Nublado	Alta	Alta	Débil	+
D_4	Lluvia	Suave	Alta	Débil	+
D_5	Lluvia	Baja	Normal	Débil	+
D_6	Lluvia	Baja	Normal	Fuerte	-
D_7	Nublado	Baja	Normal	Fuerte	+
D_8	Soleado	Suave	Alta	Débil	-
D_9	Soleado	Baja	Normal	Débil	+
D_{10}	Lluvia	Suave	Normal	Débil	+
D_{11}	Soleado	Suave	Normal	Fuerte	+
D_{12}	Nublado	Suave	Alta	Fuerte	+
D_{13}	Nublado	Alta	Normal	Débil	+
D_{14}	Lluvia	Suave	Alta	Fuerte	-

Algoritmo ID3 (ejemplo 1)







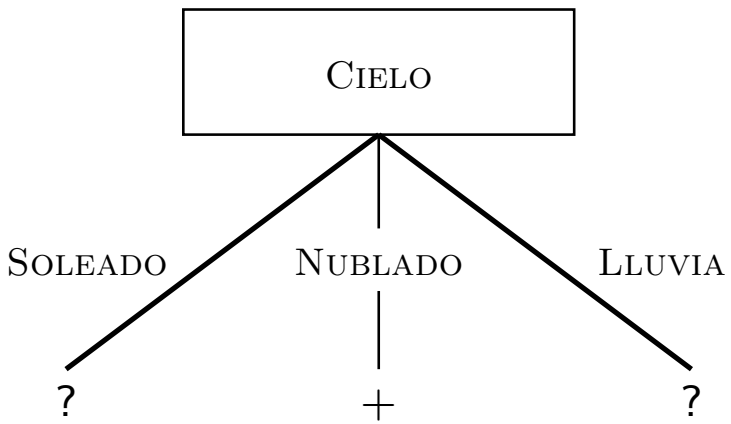


Algoritmo ID3 (ejemplo 1)

- Entropía inicial: $Ent([9^+, 5^-]) = 0{,}94$
- Selección del atributo para el nodo raíz:
  - $Ganancia(D, \text{HUMEDAD}) =$
    $0{,}94 - \frac{7}{14} \cdot Ent([3^+, 4^-]) - \frac{7}{14} \cdot Ent([6^+, 1^-]) = 0{,}151$
  - $Ganancia(D, \text{VIENTO}) =$
    $0{,}94 - \frac{8}{14} \cdot Ent([6^+, 2^-]) - \frac{6}{14} \cdot Ent([3^+, 3^-]) = 0{,}048$
  - $Ganancia(D, \text{CIELO}) =$
    $0{,}94 - \frac{5}{14} \cdot Ent([2^+, 3^-]) - \frac{4}{14} \cdot Ent([4^+, 0^-])$
    $- \frac{5}{14} \cdot Ent([3^+, 2^-]) = 0{,}246$ (mejor atributo)
  - $Ganancia(D, \text{TEMPERATURA}) =$
    $0{,}94 - \frac{4}{14} \cdot Ent([2^+, 2^-]) - \frac{6}{14} \cdot Ent([4^+, 2^-])$
    $- \frac{4}{14} \cdot Ent([3^+, 1^-]) = 0{,}02$
- El atributo seleccionado es CIELO

Algoritmo ID3 (ejemplo 1)

- Árbol parcialmente construido:



Algoritmo ID3 (ejemplo 1)

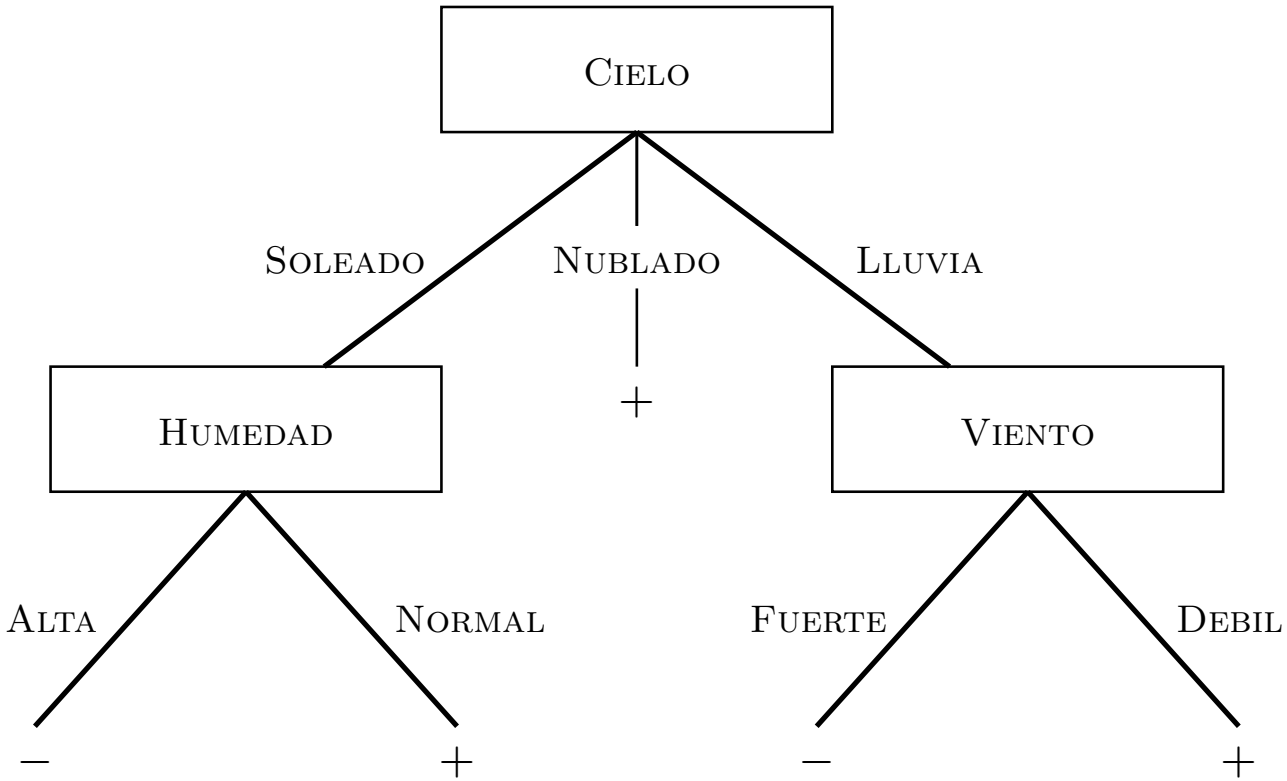
- Selección del atributo para el nodo Cielo=Soleado
- $D_{\text{Soleado}} = \{D_1, D_2, D_8, D_9, D_{11}\}$ con entropía $Ent([2^+, 3^-]) = 0{,}971$
  - $Ganancia(D_{\text{Soleado}}, \text{Humedad}) = 0{,}971 - \frac{3}{5} \cdot 0 - \frac{2}{5} \cdot 0 = 0{,}971$ (mejor atributo)
  - $Ganancia(D_{\text{Soleado}}, \text{Temperatura}) = 0{,}971 - \frac{2}{5} \cdot 0 - \frac{2}{5} \cdot 1 - \frac{1}{5} \cdot 0 = 0{,}570$
  - $Ganancia(D_{\text{Soleado}}, \text{Viento}) = 0{,}971 - \frac{2}{5} \cdot 1 - \frac{3}{5} \cdot 0{,}918 = 0{,}019$
- El atributo seleccionado es Humedad

Algoritmo ID3 (ejemplo 1)

- Selección del atributo para el nodo Cielo=Lluvia:
- $D_{\text{Lluvia}} = \{D_4, D_5, D_6, D_{10}, D_{14}\}$ con entropía $Ent([3^+, 2^-]) = 0{,}971$
  - $Ganancia(D_{\text{Lluvia}}, \text{Humedad}) = 0{,}971 - \frac{2}{5} \cdot 1 - \frac{3}{5} \cdot 0{,}918 = 0{,}020$
  - $Ganancia(D_{\text{Lluvia}}, \text{Temperatura}) = 0{,}971 - \frac{3}{5} \cdot 0{,}918 - \frac{2}{5} \cdot 1 = 0{,}020$
  - $Ganancia(D_{\text{Lluvia}}, \text{Viento}) = 0{,}971 - \frac{3}{5} \cdot 0 - \frac{2}{5} \cdot 0 = 0{,}971$ (mejor atributo)
- El atributo seleccionado es Viento

Algoritmo ID3 (ejemplo 1)

- Árbol finalmente aprendido:



Algoritmo ID3 (ejemplo 2)

- Conjunto de entrenamiento:

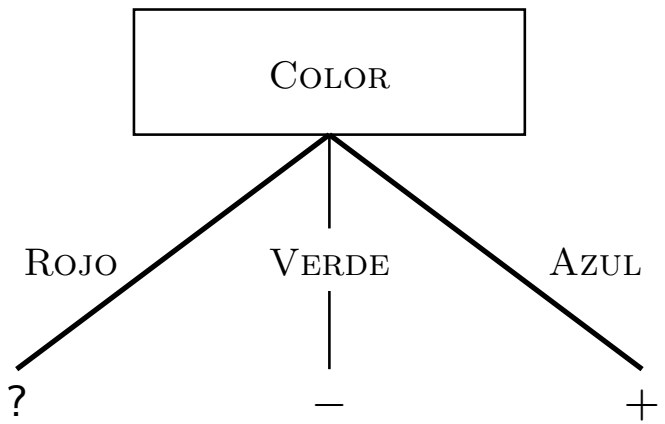
Ej.	Color	Forma	Tamaño	Clase
O_1	Rojo	Cuadrado	Grande	+
O_2	Azul	Cuadrado	Grande	+
O_3	Rojo	Redondo	Pequeño	-
O_4	Verde	Cuadrado	Pequeño	-
O_5	Rojo	Redondo	Grande	+
O_6	Verde	Cuadrado	Grande	-

Algoritmo ID3 (ejemplo 2)

- Entropía inicial en el ejemplo de los objetos, $Ent([3^+, 3^-]) = 1$
- Selección del atributo para el nodo raíz:
  - $Ganancia(D, \textsc{Color}) = 1 - \frac{3}{6} \cdot Ent([2^+, 1^-]) - \frac{1}{6} \cdot Ent([1^+, 0^-]) - \frac{2}{6} \cdot Ent([0^+, 2^-]) = 0{,}543$
  - $Ganancia(D, \textsc{Forma}) = 1 - \frac{4}{6} \cdot Ent([2^+, 2^-]) - \frac{2}{6} \cdot Ent([1^+, 1^-]) = 0$
  - $Ganancia(D, \textsc{Tamaño}) = 1 - \frac{4}{6} \cdot Ent([3^+, 1^-]) - \frac{2}{6} \cdot Ent([0^+, 2^-]) = 0{,}459$
- El atributo seleccionado es $\textsc{Color}$

Algoritmo ID3 (ejemplo 2)

- Árbol parcialmente construido:

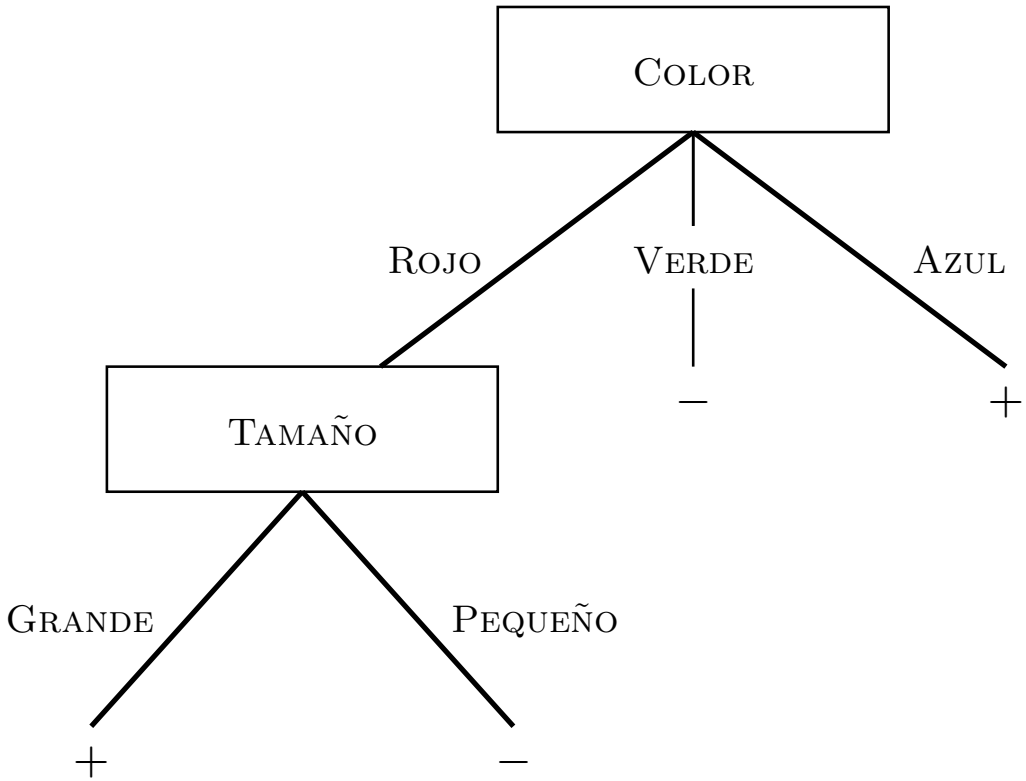


Algoritmo ID3 (ejemplo 2)

- Selección del atributo para el nodo $\textsc{Color}{=}\textsc{Rojo}$:
- $D_{\textsc{Rojo}} = \{O_1, O_3, O_5\}$ con entropía $Ent([2^+, 1^-]) = 0{,}914$
  - $Ganancia(D_{\textsc{Rojo}}, \textsc{Forma}) = 0{,}914 - \frac{1}{3} \cdot Ent([1^+, 0^-]) - \frac{2}{3} \cdot Ent([1^+, 1^-]) = 0{,}247$
  - $Ganancia(D_{\textsc{Rojo}}, \textsc{Tamaño}) = 0{,}914 - \frac{2}{3} \cdot Ent([2^+, 0^-]) - \frac{1}{3} \cdot Ent([0^+, 1^-]) = 0{,}914$
- El atributo seleccionado es $\textsc{Tamaño}$

Algoritmo ID3 (ejemplo 2)

- Árbol finalmente aprendido:



Algunos cuestiones prácticas a resolver en aprendizaje automático

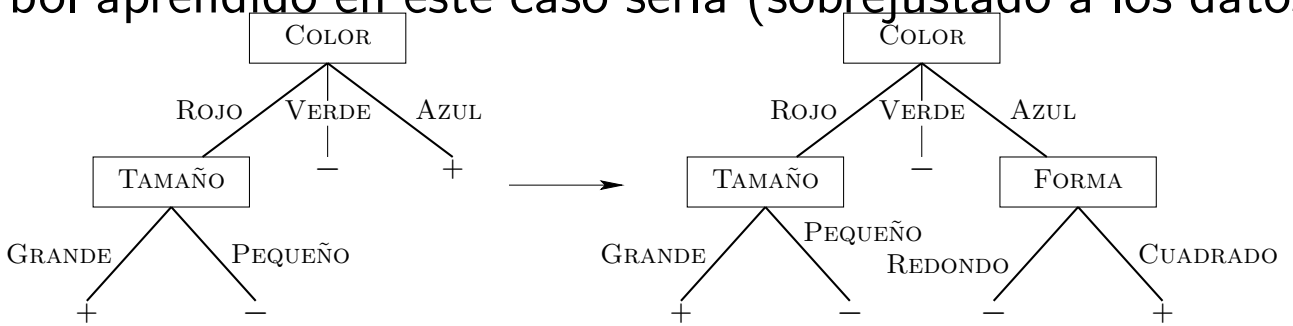
- Validar la hipótesis aprendida
  - ¿Podemos *cuantificar* la bondad de lo aprendido respecto de la explicación "real"?
- Sobreajuste
  - ¿Se ajusta *demasiado* lo aprendido a los datos concretos que se han usado en el aprendizaje?

Medida del rendimiento del aprendizaje

- Conjuntos de entrenamiento y prueba (*test*)
  - Aprender con el conjunto de entrenamiento
  - Medir el rendimiento en el conjunto de prueba:
    - proporción de ejemplos bien clasificados en el conjunto de prueba
  - Estratificación: cada clase correctamente representada en el entrenamiento y en la prueba
- A veces es necesario usar un tercer conjunto para *validar* el ajuste de parámetros del modelo
- Si no tenemos suficientes ejemplos como para apartar un conjunto de prueba: validación cruzada
  - Dividir en k partes, y hace k aprendizajes, cada uno de ellos tomando como prueba una de las partes y entrenamiento el resto. Finalmente hacer la media de los rendimientos.
  - En la práctica: validación cruzada, con $k = 10$ y estratificación

Sobreajuste

- Conjunto de *hipótesis* H: posibles modelos a aprender (por ejemplo, todos los posibles árboles de decisión)
- Una hipótesis $h \in H$ *sobreajusta* los ejemplos de entrenamiento si existe $h' \in H$ que se ajusta peor que h a los ejemplos pero actúa mejor sobre la distribución completa de instancias.
- Ejemplo: supongamos que por error, se incluye el ejemplo <Azul,Redondo,Pequeno> como ejemplo negativo (*ruido*)
  - El árbol aprendido en este caso sería (sobrejustado a los datos):



Sobreajuste

- El sobreajuste es el gran enemigo a batir en aprendizaje automático
- Causas de sobreajuste:
  - Ruido (errores en los datos)
  - Atributos que en los ejemplos presentan una aparente regularidad pero que no son relevantes en realidad
  - Conjuntos de entrenamiento pequeños
- Maneras de evitar el sobreajuste:
  - Evitar modelos excesivamente complejos (penalizar la complejidad)
  - Medir el rendimiento sobre conjuntos de validación independientes, para comprobar la capacidad de generalización de lo aprendido

Sobreajuste en árboles de decisión

- Maneras de evitar el sobreajuste en árboles de decisión:
  - Parar el desarrollo del árbol durante el aprendizaje, antes de que se ajuste perfectamente a todos los datos (*early stopping*)
  - Podar el árbol *a posteriori* (*post pruning*)
- *Early stopping* en aprendizaje de árboles:
  - No generar árboles por encima de una profundidad dada
  - Parar en nodos que corresponden con un porcentaje bajo de los datos
  - Parar en nodos en los que la mayoría de los ejemplos son de una clase
- Poda *a posteriori*, dos aproximaciones:
  - Transformación a reglas, podado de las condiciones de las reglas
  - Realizar podas directamente en el árbol
  - Las podas se producen siempre que reduzcan el error sobre un conjunto de validación

Podado de árboles

Algoritmo de poda para reducir el error

1. Dividir el conjunto de ejemplos en Entrenamiento, Validación (y Prueba)
2. Árbol=árbol obtenido por ID3 usando Entrenamiento
3. Continuar=True
4. Mientras Continuar:
   * Medida = proporción de ejemplos en conjunto de Validación correctamente clasificados por Árbol
   * Por cada nodo interior N de Árbol:
     - Podar temporalmente Árbol en el nodo N y sustituirlo por una hoja etiquetada con la clasificación mayoritaria en ese nodo
     - Medir la proporción de ejemplos correctamente clasificados en el conjunto de Validación.
   * Sea K el nodo cuya poda produce mejor rendimiento
   * Si este rendimiento es mejor que Medida, entonces Árbol = resultado de podar permanentemente Árbol en K
   * Si no, Continuar=Falso
5. Devolver Árbol (y su rendimiento sobre Prueba)

Atributos con valores continuos

- Reemplazamos los atributos continuos por atributos booleanos que se crean dinámicamente, introduciendo umbrales C.
  - A continuo
  - $A_{<C}$ booleano, toma el valor **SI** cuando el valor es menor que C y **NO** en otro caso.

Atributos con valores continuos

Temperatura	50	52	60	68	70	78	84
Clase	+	+	-	-	-	+	+

- Los candidatos a umbral C son los valores adyacentes con distinta clase
  - $56 = (52 + 60)/2$ y $74 = (70 + 78)/2$.
- Seleccionamos el umbral con máxima ganancia
- El nuevo atributo $A_{<C}$ compite con los restantes.
- El proceso se realiza a cada paso eligiendo el mejor umbral en el conjunto de entrenamiento.

Otras cuestiones prácticas del algoritmo ID3

- Extensiones del algoritmo:
  - Otras medidas para seleccionar atributos
  - Otras estimaciones de error
  - Atributos sin valores
  - Atributos con coste
- Algoritmos C4.5 y C5.0 (Quinlan)
- Algoritmo CART

Aplicaciones

Journal of Theoretical Biology 357 (2014) 21–25

Contents lists available at ScienceDirect

Journal of Theoretical Biology

journal homepage: www.elsevier.com/locate/yjtbi

Decision trees for the analysis of genes involved in Alzheimer's disease pathology

Sonia L. Mestizo Gutiérrez [a], Marisol Herrera Rivero [b], Nicandro Cruz Ramírez [c], Elena Hernández [d], Gonzalo E. Aranda-Abreu [d,*]

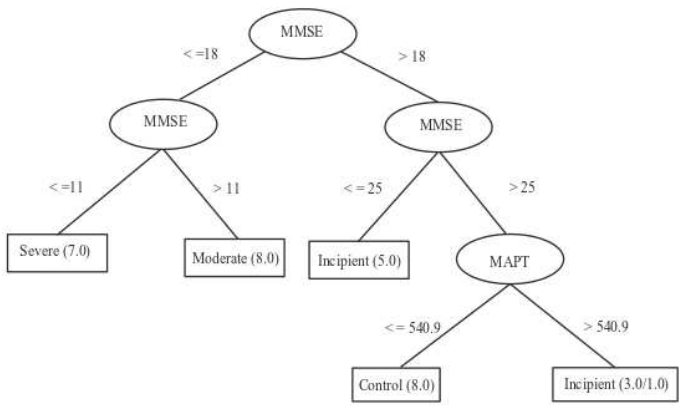
[a] Doctorado en Investigaciones Cerebrales, Universidad Veracruzana, Av. Luis Castelazo Ayala S/N, Xalapa, Veracruz 91190, Mexico
[b] Doctorado en Ciencias Biomédicas, Universidad Veracruzana, Av. Luis Castelazo Ayala S/N, Xalapa, Veracruz, Mexico
[c] Departamento de Inteligencia Artificial, Universidad Veracruzana, Sebastián Camacho 5, Centro, Xalapa, Veracruz 91000, Mexico
[d] Centro de Investigaciones Cerebrales, Cuerpo Académico de Neuroquímica, Universidad Veracruzana, Av. Luis Castelazo Ayala S/N, Xalapa, Veracruz, Mexico

Decision trees for the analysis of genes involved in Alzheimer's disease pathology Sonia L. Mestizo Gutiérrez, Marisol Herrera Rivero, Nicandro Cruz Ramírez, Elena Hernández, Gonzalo E. Aranda-Abreu. *Journal of Theoretical Biology*. Volume 357, 21 September 2014, Pages 21?25.

Aplicaciones

2.3.3. Decision trees

We tested two disease classifiers with different sets of genes, using the J48 algorithm in the Waikato Environment for Knowledge Analysis (WEKA) (Hall et al., 2009; Waikato Environment for Knowledge Analysis, 2013) an implementation of the C4.5 algorithm, with 10-fold cross-validation. The first set of genes tested consisted of those found to significantly differ by ANOVA, and the second set consisted of the AD-related genes we added to the list: *APP, APOE, BACE1, NCSTN, PSEN1, PSEN2* and *MAPT*. Expression data as well as MMSE and NFT scores were included.



Decision trees for the analysis of genes involved in Alzheimer's disease pathology Sonia L. Mestizo Gutiérrez, Marisol Herrera Rivero, Nicandro Cruz Ramírez, Elena Hernández, Gonzalo E. Aranda-Abreu. *Journal of Theoretical Biology*. Volume 357, 21 September, 2014, pages 21-25.

Aplicaciones

Diagnosis of gastric cancer using decision tree classification of mass spectral data

Yahui Su,[1]* Jing Shen,[1]* Honggang Qian,[1] Huachong Ma,[2] Jiafu Ji,[1] Hong Ma,[1] Longhua Ma,[3] Weihua Zhang,[3] Ling Meng,[1] Zhenfu Li,[1] Jian Wu,[1] Genglin Jin,[1] Jianzhi Zhang[1] and Chengchao Shou[1,4]

[1]Peking University School of Oncology and Beijing Cancer Hospital and Institute, Haidian, Beijing 100036; [2]Beijing Chaoyang Hospital, Chaoyang, Beijing 100020, China; and [3]Ciphergen Biosystems, Fremont, CA 94555, USA

(Received July 15, 2006/Revised September 2, 2006/Accepted September 4, 2006/Online publication October 19, 2006)

Cancer Science, Volume 98, Issue 1, pages 37?43, January 2007

Computers in Biology and Medicine 42 (2012) 195–204

Contents lists available at SciVerse ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/cbm

Using partial decision trees to predict Parkinson's symptoms: A new approach for diagnosis and therapy in patients suffering from Parkinson's disease

Themis P. Exarchos [a], Alexandros T. Tzallas [a], Dina Baga [a], Dimitra Chaloglou [b], Dimitrios I. Fotiadis [a,*], Sofia Tsouli [c], Maria Diakou [c], Spyros Konitsiotis [c]

[a] *Unit of Medical Technology and Intelligent Information System, Department of Materials Science and Engineering, University of Ioannina, GR 45110, Ioannina, Greece*
[b] *ANCO S.A., Athens GR 11742, Greece*
[c] *Dept. of Neurology, Medical School, University of Ioannina, GR 45110, Ioannina, Greece*

Sección 3

Sección 3
Aprendizaje de reglas

Cambiando la salida: reglas proposicionales

- Reglas de clasificación:
  - R1: **Si** Cielo=Soleado $\wedge$ Humedad=Alta
    Entonces JugarTenis= $-$
  - R2: **Si** Astigmatismo= $+$ $\wedge$ Lagrima=Normal
    Entonces Lente=Rigida
- Ventaja de usar el formalismo de reglas
  - Claridad
  - Modularidad
  - Expresividad: pueden representar cualquier conjunto de instancias
  - Métodos generalizables a primer orden de manera natural
  - Formalismo usado en sistemas basados en el conocimiento
- Reglas y árboles de decisión
  - Fácil traducción de árbol a reglas, pero no a la inversa

Aprendizaje de reglas

- Objetivo: aprender un conjunto de reglas consistente con los ejemplos
  - Una regla *cubre* un ejemplo si el ejemplo satisface las condiciones
  - Lo cubre *correctamente* si además el valor del atributo en la conclusión de la regla coincide con el valor que el ejemplo toma en ese atributo
- Una medida del ajuste de una regla R a un conjunto de ejemplos D:
  - *Frecuencia relativa*: p/t (donde $t =$ ejemplos cubiertos por R en D, $p =$ ejemplos correctamente cubiertos). Notación: $FR(R, D)$
- Algoritmos de aprendizaje de reglas:
  - ID3 + traducción a reglas
  - Cobertura
  - Algoritmos genéticos

Un conjunto de entrenamiento

Ej.	Edad	Dignostico	Astigmatismo	Lagrima	Lente
E_1	Joven	Miope	-	Reducida	Ninguna
E_2	Joven	Miope	-	Normal	Blanda
E_3	Joven	Miope	+	Reducida	Ninguna
E_4	Joven	Miope	+	Normal	Rígida
E_5	Joven	Hipermétrope	-	Reducida	Ninguna
E_6	Joven	Hipermétrope	-	Normal	Blanda
E_7	Joven	Hipermétrope	+	Reducida	Ninguna
E_8	Joven	Hipermétrope	+	Normal	Rígida
E_9	PrePresbicia	Miope	-	Reducida	Ninguna
E_{10}	PrePresbicia	Miope	-	Normal	Blanda
E_{11}	PrePresbicia	Miope	+	Reducida	Ninguna
E_{12}	PrePresbicia	Miope	+	Normal	Rígida
E_{13}	PrePresbicia	Hipermétrope	-	Reducida	Ninguna
E_{14}	PrePresbicia	Hipermétrope	-	Normal	Blanda
E_{15}	PrePresbicia	Hipermétrope	+	Reducida	Ninguna
E_{16}	PrePresbicia	Hipermétrope	+	Normal	Ninguna

Un conjunto de entrenamiento

Ej.	Edad	Dignostico	Astigmatismo	Lagrima	Lente
E_{17}	Presbicia	Miope	-	Reducida	Ninguna
E_{18}	Presbicia	Miope	-	Normal	Ninguna
E_{19}	Presbicia	Miope	+	Reducida	Ninguna
E_{20}	Presbicia	Miope	+	Normal	Rígida
E_{21}	Presbicia	Hipermétrope	-	Reducida	Ninguna
E_{22}	Presbicia	Hipermétrope	-	Normal	Blanda
E_{23}	Presbicia	Hipermétrope	+	Reducida	Ninguna
E_{24}	Presbicia	Hipermétrope	+	Normal	Ninguna

- R2: **Si** Astigmatismo=+ $\wedge$ Lagrima=Normal
  Entonces Lente=Rigida
- R2 cubre E_4, E_8, E_{12}, E_{16}, E_{20} y E_{24}, de los cuales cubre correctamente E_4, E_8, E_{12} y E_{20}

Aprendiendo reglas que cubren ejemplos

- Aprender una regla para clasificar Lente=Rigida
  - **Si** ?
    Entonces Lente=Rigida
  - Alternativas para ?, y frecuencia relativa de la regla resultante

Edad=Joven	2/8
Edad=PrePresbicia	1/8
Edad=Presbicia	1/8
Diagnostico=Miopía	3/12
Diagnostico=Hipermetropía	1/12
Astigmatismo=−	0/12
Astigmatismo=+	4/12 *
Lágrima=Reducida	0/12
Lágrima=Normal	4/12 *

- Regla parcialmente aprendida
  - **Si** Astigmatismo=+
    Entonces Lente=Rigida

Aprendiendo reglas que cubren ejemplos

- Continuamos para excluir ejemplos cubiertos incorrectamente
  - **Si** Astigmatismo=+ $\wedge$?
    Entonces Lente=Rigida
  - Alternativas para ?, y frecuencia relativa de la regla resultante

Edad=Joven	2/4
Edad=PrePresbicia	1/4
Edad=Presbicia	1/4
Diagnostico=Miopía	3/6
Diagnostico=Hipermetropía	1/6
Lágrima=Reducida	0/6
Lágrima=Normal	4/6 *

- Regla parcialmente aprendida
  - **Si** Astigmatismo=+ $\wedge$ Lagrima=Normal
    Entonces Lente=Rigida

Aprendiendo reglas que cubren ejemplos

- Continuamos para excluir ejemplos cubiertos incorrectamente
  - **Si** Astigmatismo=+ $\wedge$ Lagrima=Normal $\wedge$?
    Entonces Lente=Rigida
  - Alternativas para ?, y frecuencia relativa de la regla resultante

Edad=Joven	$2/2$ *
Edad=PrePresbicia	$1/2$
Edad=Presbicia	$1/2$
Diagnostico=Miopía	$3/3$ *
Diagnostico=Hipermetropía	$1/3$

- Regla finalmente aprendida (no cubre incorrectamente ningún ejemplo)
  - **Si** Astigmatismo=+ $\wedge$ Lagrima=Normal $\wedge$
    Diagnostico=Miopia
    Entonces Lente=Rigida

Aprendiendo reglas que cubren ejemplos

- Queda un ejemplo con Lente=Rigida no cubierto por R2
  - Comenzamos otra vez con "**Si** ? **Entonces** Lente=Rigida", pero ahora con $D' = D \setminus \{E_4, E_{12}, E_{20}\}$
- Reglas finalmente aprendidas para Lente=Rigida:
  - R1: **Si** Astigmatismo= + $\wedge$ Lagrima=Normal $\wedge$ Diagnostico=Miopia **Entonces** Lente=Rigida
  - R2: **Si** Edad=Joven $\wedge$ Astigmatismo= + $\wedge$ Lagrima=Normal **Entonces** Lente=Rigida
  - Cubren correctamente los 4 ejemplos de Lente=Rigida (y se solapan)
- Ahora se podría continuar para aprender reglas que clasifiquen:
  - Lente=Blanda
  - Lente=Ninguna

Algoritmo de aprendizaje de reglas por cobertura

Algoritmo de aprendizaje por cobertura

```
Aprendizaje-por-Cobertura(D,Atributo,v)
1. Hacer Reglas-aprendidas igual a vacío
2. Hacer E igual a D
3. Mientras E contenga ejemplos cuyo valor de Atributo es v, hacer:
3.1 Crear una regla R sin condiciones y conclusión Atributo=v
3.2 Mientras que haya en E ejemplos cubiertos por R incorrectamente
    y queden atributos que usar, hacer:
3.2.1 Elegir la MEJOR condición A=w para añadir a R, donde A es
      un atributo que no aparece en R y w es un valor de los
      posibles que puede tomar A
3.2.2 Actualizar R añadiendo la condición A=w a R
3.3 Incluir R en  Reglas-aprendidas
3.4 Actualizar E quitando los ejemplos cubiertos por R
4. Devolver Reglas-Aprendidas
```

- Algoritmo para aprender un conjunto de reglas (a partir de D)
  - Reglas para predecir situaciones en las que un Atributo dado toma un valor v

Control en el algoritmo de cobertura

- Bucle externo:
  - Añade reglas (la hipótesis se *generaliza*)
  - Cada regla añadida cubre algunos ejemplos correctamente
  - Elimina en cada vuelta los ejemplos cubiertos por la regla añadida
  - Y se añaden reglas mientras queden ejemplos sin cubrir
- Bucle interno:
  - Añade condiciones a la regla (la regla se *especializa*)
  - Cada nueva condición excluye ejemplos cubiertos incorrectamente
  - Y esto se hace mientras haya ejemplos incorrectamente cubiertos
- Cobertura frente a ID3
  - Aprende una regla cada vez, ID3 lo hace simultáneamente
  - ID3: elecciones de atributos
  - Cobertura: elcciones de parejas atributo-valor

Algoritmo de cobertura (propiedades)

- Diferentes criterios para elegir la mejor condición en cada vuelta del bucle interno:
  - Se añade la condición que produzca la regla con *mayor frecuencia relativa* (como en el ejemplo)
  - Se añade la que produzca *mayor ganancia de información*:
    $$p \cdot (log_2 \frac{p'}{t'} - log_2 \frac{p}{t})$$
    donde p'/t' es la frecuencia relativa *después* de añadir la condición y p/t es la frecuencia relativa *antes* de añadir la condición
- Las reglas aprendidas por el algoritmo de cobertura se ajustan *perfectamente* al conjunto de entrenamiento (*peligro de sobreajuste*)
  - *Early stopping*: no generar todas las condiciones
  - Podado de las reglas *a posteriori*: eliminar progresivamente condiciones hasta que no se produzca *mejora*

Sección 4

Sección 4

Aprendizaje basado en instancias: kNN

Clasificación mediante vecino más cercano

- Una técnica alternativa a construir el modelo probabilístico es calcular la clasificación directamente a partir de los ejemplos (*aprendizaje basado en instancias*)
- Idea: obtener la clasificación de un nuevo ejemplo a apartir de las categorías de los ejemplos más "cercanos".
  - Debemos manejar, por tanto, una noción de "distancia" entre ejemplos.
  - En la mayoría de los casos, los ejemplos serán elementos de R^n y la distancia, la euclídea.
  - Pero se podría usar otra noción de distancia
- Ejemplo de aplicación: clasificación de documentos

El algoritmo k-NN

- El algoritmo k-NN (de *"k nearest neighbors"*):
  - Dado un conjunto de entrenamiento (vectores numéricos con una categoría asignada) y un ejemplo nuevo
  - Devolver la categoría mayoritaria en los k ejemplos del conjunto de entrenamiento más cercanos al ejemplo que se quiere clasificar

Distancias para k-NN

- Posibles distancias usadas para definir la "cercanía":
  - Euclídea: $d_e(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
  - Manhattan: $d_m(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|$
  - Hamming: número de componentes en las que se difiere.
- La euclídea se usa cuando cada dimensión mide propiedades similares y la Mahattan en caso contrario; la distancia Hamming se puede usar aún cuando los vectores no sean numéricos.
- Normalización: cuando no todas las dimensiones son del mismo orden de magnitud, se normalizan las componentes (restando la media y dividiendo por la desviación típica)

Algunas observaciones sobre k-NN

- Elección de k:
  - Usualmente, basándonos en algún conocimiento específico sobre el problema de clasificación
  - También como resultado de pruebas en conjuntos más pequeños (conjuntos de validación)
  - Si la clasificación es binaria, preferiblemente impar, para intentar evitar empates (k=5, por ejemplo)

Variantes del algoritmo k-NN

Algoritmo k–NN con pesos

- En esta variante se consideran los k vecinos más cercanos $\{a_1, \ldots, a_k\}$ al objeto x que queremos clasificar.
- A cada uno de los k vecinos más cercamos se le asigna un peso w_i se les asigna el peso

$$w_i = \frac{1}{dist(a_i, x)}$$

- Sumamos los pesos de cada una de las posibles clasificaciones.
- El valor asignado a x será el que obtenga un mayor peso.
- Así un ejemplo a_i cuenta más cuanto más cercano esté a x.

Variantes del algoritmo k-NN

NCC (Nearest Centroid Classifier)

- Dado un conjunto entrenamiento formado por puntos de R^n junto con su clasificacion y un nuevo punto x, *NCC* asigna al nuevo punto la clasificacion de la clase cuyo centroide este mas cercano al punto.
- En otras palabras, para cada valor de clasificación calculamos el centroide de los puntos asociados y luego aplicamos k-NN sobre el conjunto de centroides con $k = 1$.

Variantes del algoritmo k-NN

k-NN con rechazo

- En esta variante, además del conjunto de entrenamiento D, el valor k y el objeto a clasificar x, necesitamos un umbral μ.
- El algoritmo toma los k puntos del conjunto de entrenamiento mas cercanos a x y devuelve el valor de clasificacion mayoritario en esos k puntos solo si el numero de puntos con esa clasificacion supera el umbral.
- Por ejemplo, si consideramos k=12 y de los 12 puntos mas cercanos hay 8 con clasificacion A, 3 con clasificacion B y 1 con clasificacion C, y el umbral es $\mu = 7$, entonces el valor de clasificacion devuelto es A. En cambio, si el umbral es $\mu = 9$, el resultado debe ser **`NO CLASIFICADO`**.

Sección 5

Sección 5

Aprendizaje de modelos probabilísticos: *Naive Bayes*

Aprendizaje de un modelo probailístico

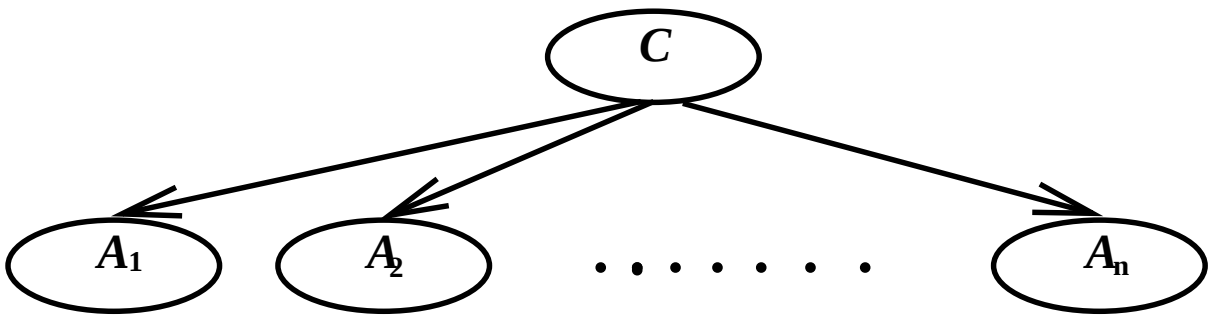
- Seguimos con el aprendizaje supervisado para clasificación:
  - Tenemos un conjunto de datos de entrenamiento ya clasificados
  - Queremos *generalizar* y ser capaces de clasificar nuevos datos
- El resultado del aprendizaje va a ser una distribución de probabilidad
  - En concreto, una red bayesiana muy simple
- Con esa red bayesiana, podremos clasificar nuevos ejemplos
  - Calculando la probabilidad de pertenecer a cada clase

Clasificadores *Naive Bayes*

- Supongamos un conjunto de atributos $A_1, \ldots, A_n$ cuyos valores determinan un valor en un conjunto finito C de posibles clases
- Tenemos un conjunto de entrenamiento D con una serie de tuplas de valores concretos para los atributos, junto con su clasificación
- Queremos aprender un clasificador tal que clasifique nuevas instancias $\langle a_1, \ldots, a_n \rangle$

La suposición *Naive*

- En el modelo probabilístico, tanto los atributos como el valor de clasificación se consideran *variables aleatorias*
- *Supondremos* un modelo *naive*:
  - Asumiremos que **los atributos son condicionalmente independientes entre sí, dado el valor de clasificación**
- Esto es lo mismo que asumir que la distribución de probabilidad conjunta de todas las variables se puede representar con la siguiente red bayesiana:



- Esta suposición normalmente o es incierta y demasiado fuerte
  - Pero a pesar de eso, el clasificador funciona bastante bien

Clasificadores *Naive Bayes*

- El valor de clasificación asignado a una nueva instancia $\langle a_1, \ldots, a_n \rangle$, notado c_{NB} vendrá dado por

$$c_{NB} = \underset{c_j \in V}{argmax}\, P(c_j | a_1, \ldots, a_n)$$

- Aplicando las independencias condicionales asumidas en la red bayesiana tenemos:

$$P(C | a_1, \ldots, a_n) = \alpha \cdot P(C, a_1, \ldots, a_n) = \alpha \cdot P(C) \prod_i P(a_i | C)$$

- Si estamos interesados en conocer únicamente la clase más probable, podemos ignorar la constante de normalización, y entonces:

$$c_{NB} = \underset{c_j \in C}{argmax}\, P(c_j) \prod_i P(a_i | c_j)$$

Estimación de probabilidades *Naive Bayes*

- Para el proceso de aprendizaje, sólo tenemos que estimar las probabilidades $P(c_j)$ (probabilidades *a priori*) y $P(a_i|c_j)$ (probabilidades *condicionadas*). Son muchas menos que en el caso general.
- Mediante cálculo de sus frecuencias en el conjunto de entrenamiento, obtenemos estimaciones de *máxima verosimilitud* de esas probabilidades:

$$P(c_j) = \frac{n(C = c_j)}{N} \qquad P(a_i|c_j) = \frac{n(A_i = a_i, C = c_j)}{n(C = c_j)}$$

  donde N es el número total de ejemplos, $n(C = c_j)$ es el número de ejemplos clasificados como c_j y $n(A_i = a_i, C = c_j)$ es el número de ejemplos clasificados como c_j cuyo valor en el atributo A_i es a_i.
- A pesar de su aparente sencillez, los clasificadores Naive Bayes tienen un rendimiento comparable al de los árboles de decisión, las reglas o las redes neuronales

Clasificador Naive Bayes: un ejemplo

Ej.	Cielo	Temperatura	Humedad	Viento	JugarTenis
D_1	Soleado	Alta	Alta	Débil	-
D_2	Soleado	Alta	Alta	Fuerte	-
D_3	Nublado	Alta	Alta	Débil	+
D_4	Lluvia	Suave	Alta	Débil	+
D_5	Lluvia	Baja	Normal	Débil	+
D_6	Lluvia	Baja	Normal	Fuerte	-
D_7	Nublado	Baja	Normal	Fuerte	+
D_8	Soleado	Suave	Alta	Débil	-
D_9	Soleado	Baja	Normal	Débil	+
D_{10}	Lluvia	Suave	Normal	Débil	+
D_{11}	Soleado	Suave	Normal	Fuerte	+
D_{12}	Nublado	Suave	Alta	Fuerte	+
D_{13}	Nublado	Alta	Normal	Débil	+
D_{14}	Lluvia	Suave	Alta	Fuerte	-

Clasificador Naive Bayes: un ejemplo

- Supongamos que queremos predecir si un día soleado, de temperatura suave, humedad alta y viento fuerte es bueno para jugar al tenis
- Según el clasificador Naive Bayes:

$$c_{NB} = \underset{c_j \in \{+,-\}}{argmax}\ P(c_j)P(soleado|c_j)P(suave|c_j)P(alta|c_j)P(fuerte|c_j)$$

- Así que necesitamos estimar todas estas probabilidades, lo que hacemos simplemente calculando frecuencias en la tabla anterior:
  - $p(+) = 9/14$, $p(-) = 5/14$, $p(soleado|+) = 2/9$, $p(soleado|-) = 3/5$, $p(suave|+) = 4/9$, $p(suave|-) = 2/5$, $p(alta|+) = 3/9$, $p(alta|-) = 4/5$, $p(fuerte|+) = 3/9$ y $p(fuerte|-) = 3/5$

Clasificador Naive Bayes: un ejemplo

- Por tanto, las dos probabilidades (sin normalizar) a posteriori son:
  - $P(+)P(soleado|+)P(suave|+)P(alta|+)P(fuerte|+) = 0{,}0070$
  - $P(-)P(soleado|-)P(suave|-)P(alta|-)P(fuerte|-) = 0{,}0411$
- Así que el clasificador devuelve la clasificación con mayor probabilidad a posteriori, en este caso la respuesta es $-$ (no es un día bueno para jugar al tenis)

Detalles técnicos sobre las estimaciones: log-probabilidades

- Tal y como estamos calculando las estimaciones, existe el riesgo de que algunas de ellas sean excesivamente bajas
- Si realmente alguna de las probabilidades es baja y tenemos pocos ejemplos en el conjunto de entrenamiento, lo más seguro es que la estimación de esa probabilidad sea 0
- Esto plantea dos problemas:
  - La inexactitud de la propia estimación
  - Afecta enormemente a la clasificación que se calcule, ya que se multiplican las probabilidades estimadas y por tanto si una de ellas es 0, anula a las demás
- Una primera mejora técnica, intentado evitar productos muy bajos: usar logaritmos de las probabilidades.
  - Los productos se transforman en sumas

$$c_{NB} = \underset{c_j \in C}{argmax}[log(P(c_j)) + \sum_i log(P(a_i|c_j))]$$

Detalles técnicos sobre las estimaciones: suavizado

- Problema en la estimaciones:
  - Probabilidades nulas o muy bajas, por ausencia en el conjunto de entrenamiento de algunos valores de atributos en algunas categorías
  - Sobreajuste
- Idea: *suponer* que tenemos m ejemplos adicionales, cuyos valores se distribuyen teóricamente *a priori* de alguna manera.
- Estimación *suavizada* de una probabilidad, a partir de observaciones:

$$\frac{n' + m \cdot p}{n + m}$$

  - n' y n son, respectivamente, el número de ejemplos favorables y totales observados
  - p es una estimación *a priori* de la probabilidad que se quiere estimar.
  - m es una constante (llamada *tamaño de muestreo equivalente*) que indica el número de ejemplos adicionales (ficticios)

Suavizado aditivo (o de Laplace)

- Un caso particular de lo anterior se suele usar para la estimación de las probabilidades condicionales en Naive Bayes:

$$P(a_i|c_j) = \frac{n(A_i = a_i, C = c_j) + k}{n(C = c_j) + k|A_i|}$$

  donde k es un número fijado y $|A_i|$ es el número de posibles valores del atributo $|A_i|$.
- Intuitivamente: se supone que además de los del conjunto de entrenamiento, hay k ejemplos en la clase c_j por cada posible valor del atributo A_i
- Usualmente $k = 1$, pero podrían tomarse otros valores
  - Elección de k: experimentando con los dstintos rendimientos sobre un *conjunto de validación*

Sección 6

Sección 6
Clustering

Clustering

. . . in cluster analysis a group of objects is split up into a number of more or less homogeneous subgroups on the basis of an often subjectively chosen meausure of similarity (i.e., chosen subjectively based on its ability to create "interesting" clusters), such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to different subgroups.
(Backer & Jain, 1981)

Clustering

- Se trata de dividir un conjunto de datos de entrada en subconjuntos (*clusters*), de tal manera que los elementos de cada subconjunto compartan cierto patrón o características a priori desconocidas
- Aprendizaje *no supervisado*: no tenemos información sobre qué cluster corresponde a cada dato.
- Aplicaciones de clustering:
  - Minería de datos
  - Procesamiento de imágenes digitales
  - Bioinformática
- Tipos:
  - Clustering de partición estricta
  - Clustering jerárquico
  - Clustering basado en densidad

Clustering

Clustering de partición estricta

Dado un conjunto de ejemplos $D = \{\vec{x_1}, \ldots, \vec{x_j}, \ldots, \vec{x_N}\}$ con $\vec{x_j} = (x_{j1}, \ldots, x_{jd}\} \in \mathbb{R}^d$, el **clustering de partición estricta** *Hard partitional clustering* busca una partición de D en K clusters, $\mathcal{P} = \{C_1, \ldots, C_K\}$ con $K \leq N$ tal que

- $C_i \neq \emptyset$ para $i \in \{1, \ldots, K\}$
- $\bigcup_{i=1}^{i=K} C_i = D$
- $C_i \cap C_j = \emptyset$ para todo $i, j \in \{1, \ldots, K\}$ con $i \neq j$.

Clustering

Clustering jerárquico

Dado un conjunto de ejemplos $D = \{\vec{x_1}, \ldots, \vec{x_j}, \ldots, \vec{x_N}\}$ con $\vec{x_j} = (x_{j1}, \ldots, x_{jd}\} \in \mathbb{R}^n$, el **clustering jerárquico** *Hierarchical clustering* busca construir un conjunto de particiones anidadas de D con estructura de árbol, $\mathcal{H} = \{P_1, \ldots, P_Q\}$ con $Q \leq N$ (donde cada P_j es una partición) tal que si $C_i \in P_m$ y $C_j \in P_l$ con $m > l$ entonces $C_i \subset C_j$ o $C_i \cap C_j = \emptyset$, para todo $i, j, m, l \in \{1, \ldots, Q\}$, $i \neq j$. Cada P_i es una partición de D.

Clustering

- Como hemos visto, el clustering de partición estricta divide al conjunto de datos en clusters que no tienen ninguna estuctura interna.
- En cambio, el clusteing jerárquico da estructura interna a los clusters. De hecho, de manera recursiva, cada cluster está dividido en clusters internos, en una estructura anidada que va desde un cluster general conteniendo a todos los individuos, hasta clusters que contienen un único elemento.

El concepto de distancia

- La idea básica del *clustering* consiste en agrupar las instancias según su *proximidad*, esto es, dos instancias pertenecerán al mismo cluster si están *próximas* y pertenecerán a clusters distintos si están *lejanas*.
- Distancias:
  - Euclídea: $d_e(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
  - Manhattan: $d_m(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|$
  - Hamming: número de componentes en las que se difiere.
- Problemas:
- Incompatibilidad de las unidades de medida: Metros, gramos, litros,. . . (Normalización); Tipos de variables: Booleanas, nominales, numéricas, . . . ; Datos incompletos . . .

Dos ejemplos

- *Color quantization*:
  - Una imagen digital almacenada con 24 bits/pixel (aprox. 16 millones de colores) se tiene que mostrar sobre una pantalla que sólo tiene 8 bits/pixel (256 colores)
  - ¿Cuál es la mejor correspondencia entre los colores de la imagen original y los colores que pueden ser mostrados en la pantalla?
- Mezcla de distribuciones:
  - Tenemos una serie de datos con el peso de personas de un pais; no tenemos información sobre si el peso viene de un varón o de una mujer, pero sabemos que la distribución de pesos es de tipo normal, y que en los hombres es distinta que en las mujeres
  - Atendiendo a los datos, ¿podemos *aprender* de qué dos distribuciones de probabilidad vienen?

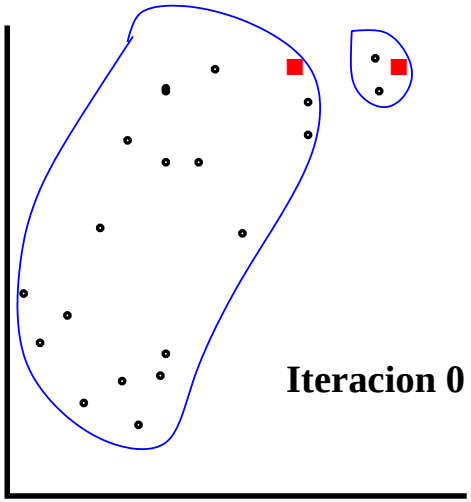
Un algoritmo de partición estricta: k-medias

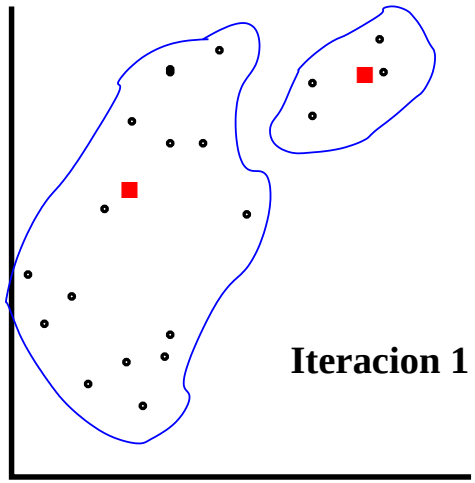
- Entrada: un número k de clusters, un conjunto de datos $\{x_i\}_{i=1}^{N}$ y una función de distancia
- Salida: un conjunto de k centros $m_1, \ldots, m_k$

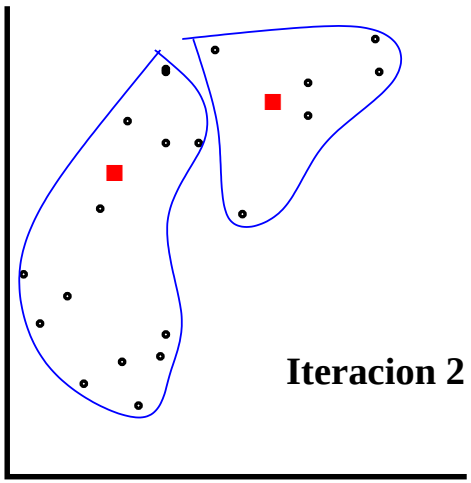
k-medias(k,datos,distancia)

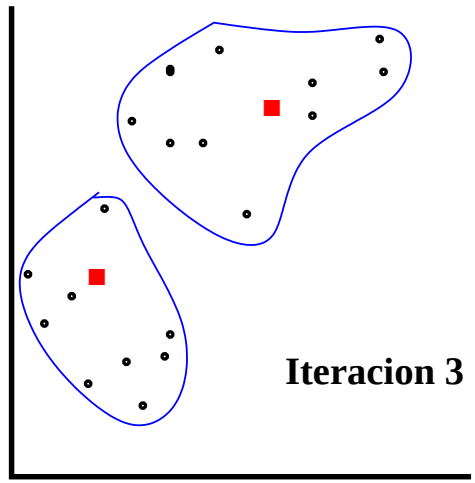
```
1. Inicializar m_i (i=1,...,k) (aleatoriamente o con algún
   criterio heurístico)
2. REPETIR (hasta que los m_i no cambien):
   2.1 PARA j=1,...,N, HACER:
       Calcular el cluster correspondiente a x_j, escogiendo,
       de entre todos los m_i, el m_h tal que
       distancia(x_j,m_h) sea mínima
   2.2 PARA i=1,...,k HACER:
       Asignar a m_i la media aritmética de los datos
       asignados al cluster i-ésimo
3. Devolver m_1,...,m_n
```

Idea gráfica intuitiva en el algoritmo de k-medias









Ejemplo en el algoritmo k-medias

- Datos sobre pesos de la población: 51, 43, 62, 64, 45, 42, 46, 45, 45, 62, 47, 52, 64, 51, 65, 48, 49, 46, 64, 51, 52, 62, 49, 48, 62, 43, 40, 48, 64, 51, 63, 43, 65, 66, 65, 46, 39, 62, 64, 52, 63, 64, 48, 64, 48, 51, 48, 64, 42, 48, 41
- El algoritmo, aplicado con $k = 2$ y distancia euclídea, encuentra dos centros $m_1 = 63{,}63$ y $m_2 = 46{,}81$ en tres iteraciones
- 19 datos pertenecen al primer cluster y 32 al segundo cluster

Diversas cuestiones sobre el algoritmo k-medias

- Inicialización: aleatoria o con alguna técnica heurística (por ejemplo, partir los datos aleatoriamente en k clusters y empezar con los centros de esos clusters)
- En la práctica, los centros con los que se inicie el algoritmo tienen un gran impacto en la calidad de los resultados que se obtengan

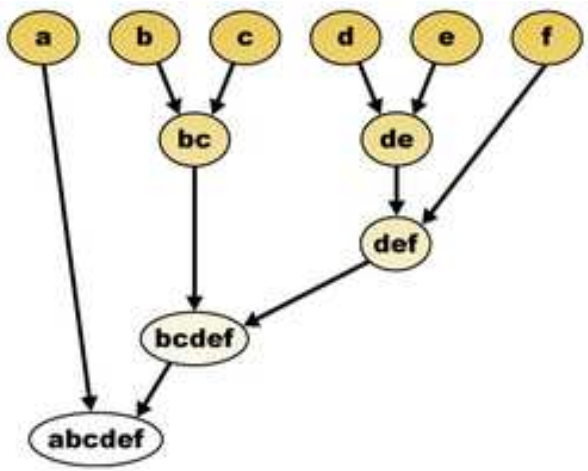
Clustering jerárquico

Dependiendo del orden en que se cree la jerarquía de clusters, los algoritmos de clustering jerárquico se dividen en dos tipos:

- Clustering jerárquico aglomerativo: Partimos de clusters conteniendo un único ejemplo y vamos agrupando los clusters, obteniendo agrupamientos cada vez de mayor tamaño hasta obtener un cluster final con todos los individuos.
- Clustering jerárquico divisor: Empezamos con un cluster con todos los individuos y vamos realizando particiones de los clusters obtenidos hasta obtener clusters conteniendo un único ejemplo.

Clustering jerárquico

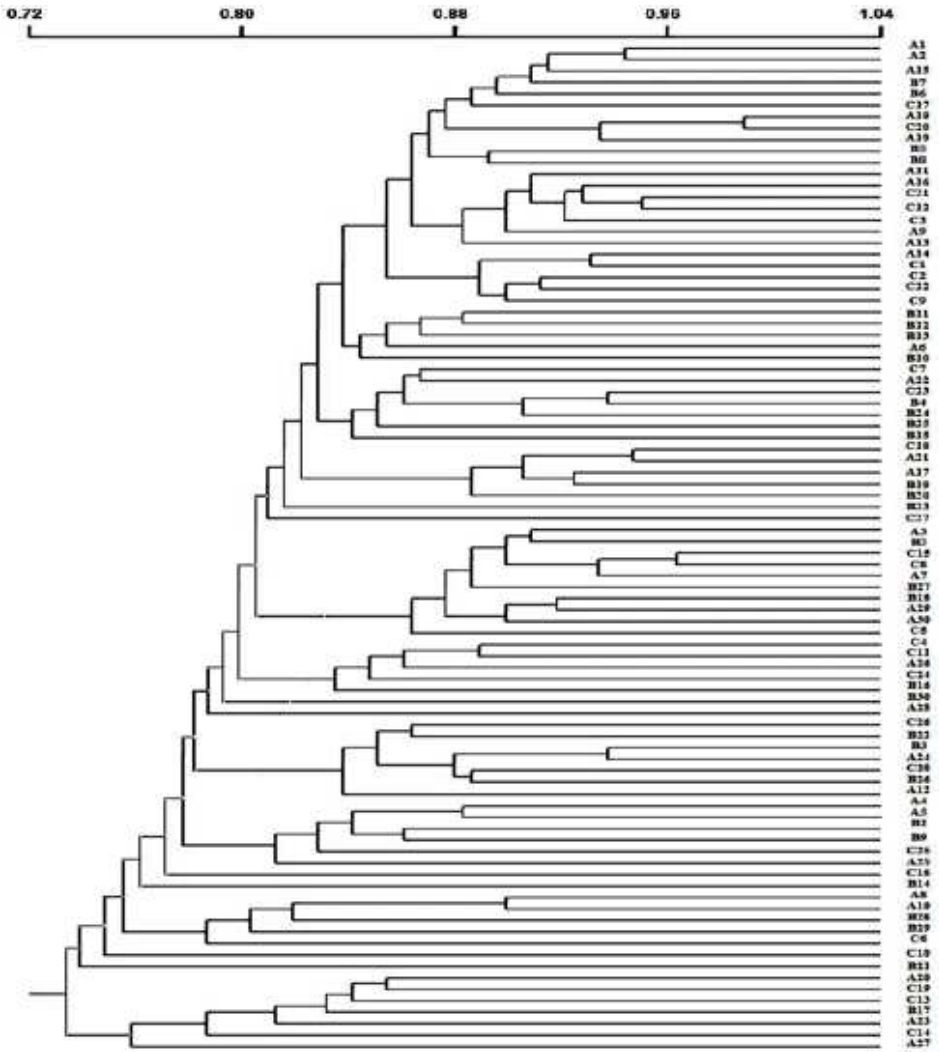
Con independencia de que se use clustering aglomerativo o divisor, los resultados suelen representarse con una estuructura de árbol llamada *dendrodrama*



La raíz representa el conjunto completo y cada una de las hojas representa una instancia.

Clustering jerárquico

Ejemplo de dendrograma



Clustering jerárquico

Complejidad

Consideremos un problema de clustering jerárquico con N puntos

- Los métodos de *Clustering jerárquico divisor* tienen que considerar $2^{N-1} - 1$ posibles maneras de dividir el conjunto inicial en dos subconjuntos.
- Los métodos de *Clustering jerárquico aglomerativo* deben considerar la distancia entre pares de puntos de orden $O(N^2)$.
- Por tanto *Clustering jerárquico aglomerativo* se usan mucho más que los de tipo *divisor*.

Clustering jerárquico aglomerativo

El esquema general de *Clustering jerárquico aglomerativo* es el siguiente:

1. Inicializamos el algoritmo con N clusters individuales. Calculamos la matriz de proximidad (basada en alguna definición de distancia) para los N clusters.
2. En la matriz de proximidad, buscamos la menor distancia entre clusters. Según definamos la *distancia entre clusters*, tendremos diferentes algoritmos. Combinamos en un único cluster aquellos que estén a distancia mínima.
3. Actualizamos la matriz de proximidad considerando los nuevos clusters.
4. Repetimos los pasos 2 y 3 hasta que quede un único cluster.

Aplicaciones

PLoS One. 2017; 12(2): e0171429. PMCID: PMC5325197
Published online 2017 Feb 24. doi: 10.1371/journal.pone.0171429

Clustering cancer gene expression data by projective clustering ensemble

Xianxue Yu, Guoxian Yu, and Jun Wang*

Guy N Brock, Editor

Abstract

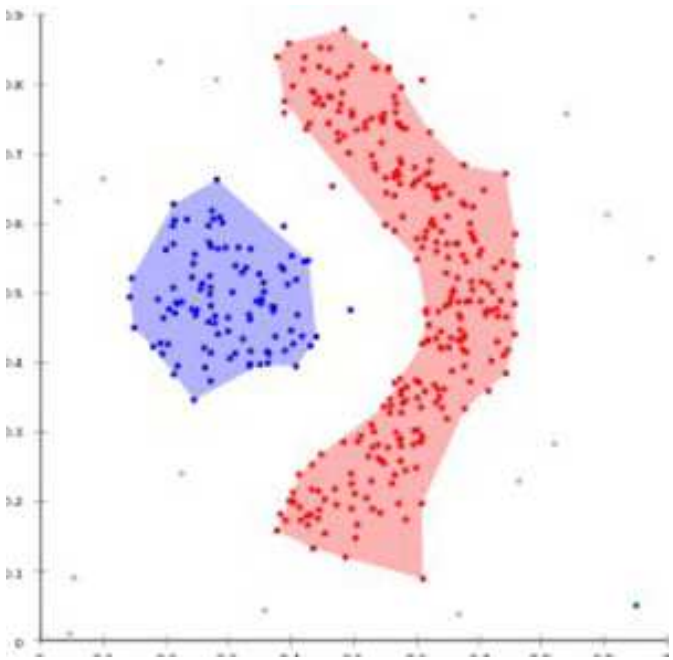
Gene expression data analysis has paramount implications for gene treatments, cancer diagnosis and other domains. Clustering is an important and promising tool to analyze gene expression data. Gene expression

Clustering cancer gene expression data by projective clustering ensemble Xianxue Yu, Guoxian Yu and Jun Wang

PLoS One. 2017; 12(2): doi: 10.1371/journal.pone.0171429

Clustering basado en densidad

- Basados en la *densidad* de puntos en distintas regiones.
- No necesitamos dar *a priori* el número de clusters como en k-medias
- Puede generar clusters de formas arbitrarias.



DBSCAN

Density Based Spatial Clustering of Applications with Noise

El algoritmo necesita dos parámetros

- *minPts* que es un entero positivo (número mínimo de puntos)
- ϵ que es un real positivo (distancia)

Los puntos los clasificamos como

- Un punto p es núcleo si hay al menos *minPts* puntos a distancia menor que ϵ de p.
- Un punto q es frontera si hay menos de *minPts* puntos a distancia menor que ϵ de q, pero hay al menos un punto p a distancia menor que ϵ de q cumpliendo que p es núcleo.
- Un punto r es ruido si no es núcleo ni frontera.

DBSCAN

Density Based Spatial Clustering of Applications with Noise

- Se dice que q es directamente alcanzable desde p si p es núcleo y la distancia entre p y q es menor que ϵ.
- Un punto q es alcanzable desde p si existe una secuencia de puntos $p_1, \ldots, p_n$ donde $p_1 = p$ y $p_n = q$ tal que cada punto p_{i+1} es directamente alcanzable desde p_i; es decir, todos los puntos de la secuencia deben ser puntos núcleos, con la posible excepción de q

DBSCAN

Density Based Spatial Clustering of Applications with Noise

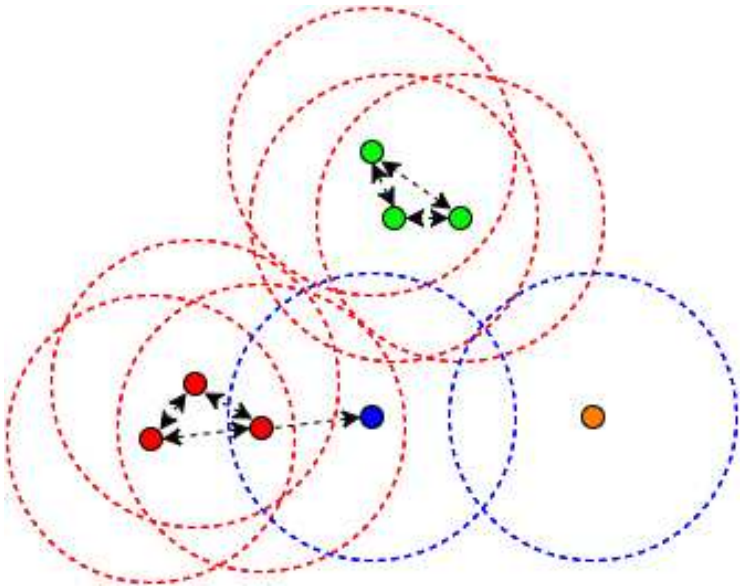
Algoritmo básico de DBSCAN

Entrada: Nube de puntos D, ϵ, *minPts*

1. Determinar los puntos núcleo, frontera y ruido en función de ϵ y *minPts*.
2. Crear un grafo conectando dos puntos núcleo si están a distancia menor que ϵ.
3. Conectar los puntos frontera a uno de los puntos núcleo que estén a distancia menor que ϵ (no determinista)
4. Devolver los puntos de cada componente conexa como un cluster.

DBSCAN

Density Based Spatial Clustering of Applications with Noise



Ejemplo de salida de DBSCAN con $minPts = 2$. Se han obtenido dos clusters, uno con tres puntos y otro con cuatro. Los puntos rojos y verdes son puntos núcleo, el punto azul es frontera y el punto naranja es ruido.

Bibliografía

- Mitchell, T.M. *Machine Learning* (McGraw-Hill, 1997)
  - Caps. 3,6,8 y 10
- Russell, S. y Norvig, P. *Artificial Intelligence (A Modern Approach)* (3rd edition) (Prentice Hall, 2010)
  - Seccs. 18.1, 18.2, 18.3, 20.1 y 20.2
- Witten, I.H. y Frank, E. *Data mining* (Third edition) (Morgan Kaufmann Publishers, 2011)
  - Cap. 3, 4, 5 y 6.
- Alpaydin, E. *Introduction to Machine Learning* (third edition) (The MIT Press, 2014)
- Xu, R y Wunsch II, D.C. *Clustering* (IEEE Press, 2009)