Árboles de decisión

Christopher Expósito Izquierdo
Airam Expósito Márquez
Israel López Plata
Belén Melián Batista
J. Marcos Moreno Vega

{**cexposit, aexposim, ilopezpl, mbmelian, jmmoreno**}**@ull.edu.es**

Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna

Índice

INTRODUCCIÓN

Clasificación (i)

- La tarea de clasificación consiste en asignar objetos a una de las clases previamente definidas.
- Se trata de una tarea presente en multitud de aplicaciones:
  - detectar correo spam (spam, no spam)
  - clasificar células tumorales (benignas, malignas)
  - clasificar créditos bancarios (conceder, denegar)

Clasificación (ii)

- Para llevar acabo la tarea de clasificación se dispone de un conjunto de objetos caracterizados por el par de atributos (x, y), donde x es un vector de características e y es la conocida como etiqueta de clase.
- A la etiqueta de clase y se le conoce también como categoría.

Algoritmo de Hunt

Datos disponibles

		Atributos					Clase
		x_1	x_2	$\cdots$	x_i	$\cdots$	x_n	y
Objetos	id_1	x_1^1	x_2^1	$\cdots$	x_i^1	$\cdots$	x_n^1	y^1
	id_2	x_1^2	x_2^2	$\cdots$	x_i^2	$\cdots$	x_n^2	y^2
	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	id_j	x_1^j	x_2^j	$\cdots$	x_i^j	$\cdots$	x_n^j	y^j
	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	id_m	x_1^m	x_2^m	$\cdots$	x_i^m	$\cdots$	x_n^m	y^m

Clasificación (ii)

- **Definición.**- La tarea de clasificación consiste en aprender una función f que asocia a cada vector x una de las clases predefinidas y.
- A la función f se la conoce también como modelo de clasificación.
- El modelo de clasificación puede adoptar diferentes formas: árbol de decisión, reglas, red neuronal ...
- El vector x puede tomar valores nominales o numéricos, pero la etiqueta y es siempre nominal. Cuando la variable y es numérica se construye un árbol de regresión.

Utilidad de la clasificación

- **Modelo descriptivo.** Puede servir para distinguir entre objetos de diferentes clases identificando las características que las describen.
- **Modelo predictivo.** Puede usarse para predecir la clase a la que pertenece un objeto conocidas sus características. Es el uso que habitualmente se le da a la clasificación.

Ejemplo 1

ID	PREVISIÓN	TEMPERATURA	HUMEDAD	VIENTO	JUGAR
id_1	Soleado	Alta	Alta	Débil	No
id_2	Soleado	Alta	Alta	Fuerte	No
id_3	Nuboso	Alta	Alta	Débil	Sí
id_4	Lluvioso	Media	Alta	Débil	Sí
id_5	Lluvioso	Fría	Normal	Débil	Sí
id_6	Lluvioso	Fría	Normal	Fuerte	No
id_7	Nuboso	Fría	Normal	Fuerte	Sí
id_8	Soleado	Media	Alta	Débil	No
id_9	Soleado	Alta	Normal	Débil	Sí
id_{10}	Lluvioso	Media	Normal	Débil	Sí
id_{11}	Soleado	Media	Normal	Fuerte	Sí
id_{12}	Nuboso	Media	Alta	Fuerte	Sí
id_{13}	Nubos	Alta	Normal	Débil	Sí
id_{14}	Lluvioso	Fría	Alta	Fuerte	No

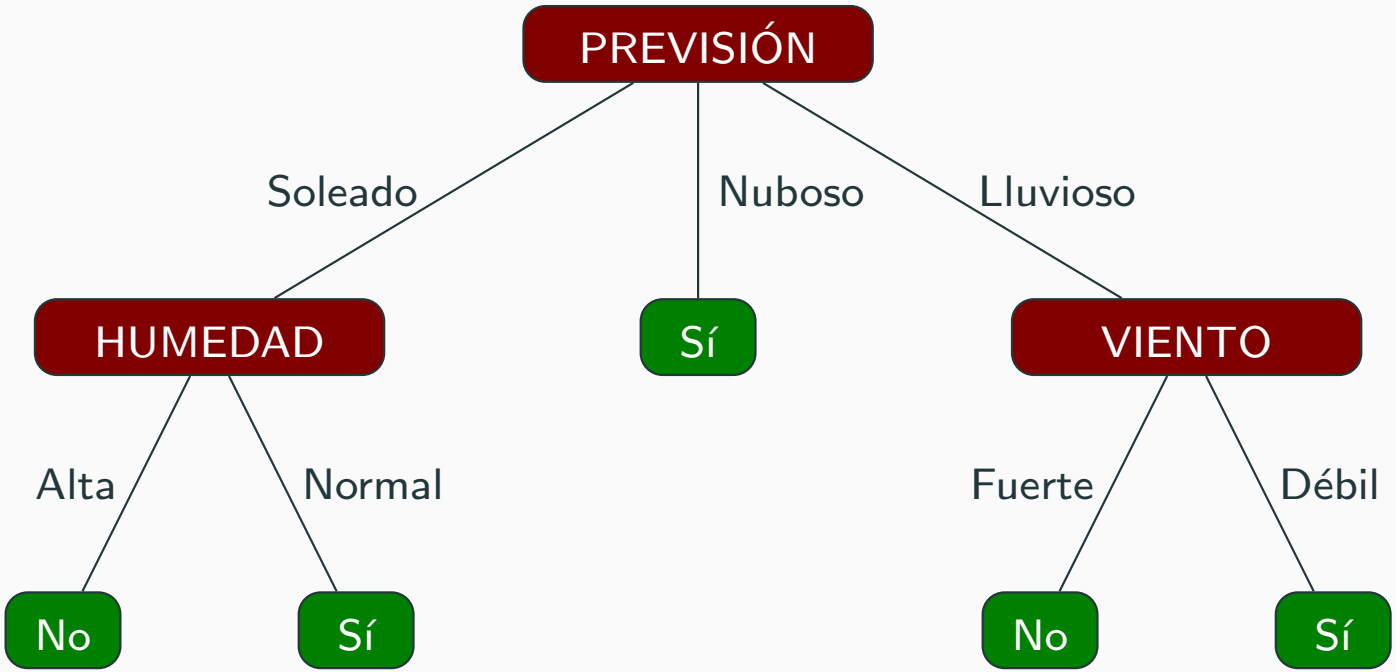
Ejemplo 2

ID	CASA	ESTADO	INGRESOS	PRÉSTAMO
id_1	Propiedad	Soltero	125000	Conceder
id_2	Alquiler	Casado	100000	Conceder
id_3	Alquiler	Soltero	70000	Conceder
id_4	Propiedad	Casado	12000	Conceder
id_5	Alquiler	Divorciado	95000	Denegar
id_6	Alquiler	Casado	60000	Conceder
id_7	Propiedad	Divorciado	220000	Conceder
id_8	Alquiler	Soltero	85000	Denegar
id_9	Alquiler	Casado	75000	Conceder
id_{10}	Alquiler	Soltero	90000	Conceder

¿QUÉ SON Y CÓMO SE EMPLEAN?

Árbol de decisión

Ejemplo 1



Árbol de decisión

Ejemplo 2

Árbol de decisión

Descripción y uso

- Cada nodo del árbol se corresponde con un atributo y de él parten tantas ramas como valores distintos tiene ese atributo.
- En las hojas del árbol se encuentran todos o algunos de los valores de la variable clase.
- Dada un árbol de decisión, para clasificar una nueva instancia se inspecciona el mismo desde la raíz hasta llegar a un nodo hoja.
- Cada nodo representa un test sobre un atributo y el valor correspondiente en la instancia indica la rama del árbol que debe recorrerse. El proceso se repite hasta alcanzar un nodo hoja. El valor de ese nodo suministra la clase a la que pertenece la instancia.

Árbol de decisión

Disyunciones de conjunciones (reglas)

- En general los árboles de decisión representan disyunciones de conjunciones de los valores de los atributos.
- Cada rama del árbol es una conjunción y el árbol en su conjunto una disyunción de esas conjunciones.

Árbol de decisión

Disyunciones de conjunciones (reglas)

- El árbol del ejemplo anterior se corresponde con el siguiente conjunto de reglas:
  - **IF** (Previsión = Soleado) **and** (Humedad = alta) **THEN** (Jugar = No)
  - **IF** (Previsión = Soleado) **and** (Humedad = normal) **THEN** (Jugar = Sí)
  - **IF** (Previsión = Nuboso) **THEN** (Jugar = Sí)
  - **IF** (Previsión = Lluvioso) **and** (Viento = Fuerte) **THEN** (Jugar = No)
  - **IF** (Previsión = Lluvioso) **and** (Viento = Débil) **THEN** (Jugar = Sí)

Árbol de decisión

Problemas apropiados

- **Instancias representadas por pares atributo valor.** Los árboles son apropiados cuando cada atributo toma un número pequeño de valores.
- **La función objetivo toma valores discretos.** Aunque también existen algoritmos que permiten construir árboles de decisión cuando la variable de salida es continua.
- **Los datos de entrenamiento pueden contener errores.** Los algoritmos para construir árboles son robustos a errores de clasificación de los ejemplos de entrenamiento y a errores en los valores de los atributos.
- **Los datos de entrenamiento pueden contener valores desconocidos en algunos atributos.** Pueden construirse cuando algunos ejemplos de entrenamiento tienen valores desconocidos en algunos de los atributos.

Problemas apropiados

- **Instancias representadas por pares atributo valor.** Los árboles son apropiados cuando cada atributo toma un número pequeño de valores.
- **La función objetivo toma valores discretos.** Aunque también existen algoritmos que permiten construir árboles de decisión cuando la variable de salida es continua.
- **Los datos de entrenamiento pueden contener errores.** Los algoritmos para construir árboles son robustos a errores de clasificación de los ejemplos de entrenamiento y a errores en los valores de los atributos.
- **Los datos de entrenamiento pueden contener valores desconocidos en algunos atributos.** Pueden construirse cuando algunos ejemplos de entrenamiento tienen valores desconocidos en algunos de los atributos.

Problemas apropiados

- **Instancias representadas por pares atributo valor.** Los árboles son apropiados cuando cada atributo toma un número pequeño de valores.
- **La función objetivo toma valores discretos.** Aunque también existen algoritmos que permiten construir árboles de decisión cuando la variable de salida es continua.
- **Los datos de entrenamiento pueden contener errores.** Los algoritmos para construir árboles son robustos a errores de clasificación de los ejemplos de entrenamiento y a errores en los valores de los atributos.
- **Los datos de entrenamiento pueden contener valores desconocidos en algunos atributos.** Pueden construirse cuando algunos ejemplos de entrenamiento tienen valores desconocidos en algunos de los atributos.

Problemas apropiados

- **Instancias representadas por pares atributo valor.** Los árboles son apropiados cuando cada atributo toma un número pequeño de valores.
- **La función objetivo toma valores discretos.** Aunque también existen algoritmos que permiten construir árboles de decisión cuando la variable de salida es continua.
- **Los datos de entrenamiento pueden contener errores.** Los algoritmos para construir árboles son robustos a errores de clasificación de los ejemplos de entrenamiento y a errores en los valores de los atributos.
- **Los datos de entrenamiento pueden contener valores desconocidos en algunos atributos.** Pueden construirse cuando algunos ejemplos de entrenamiento tienen valores desconocidos en algunos de los atributos.

ALGORITMOS DE INDUCCIÓN

Algoritmo de Hunt

Descripción

Sean S_t el conjunto de ejemplos asociados con el nodo t y $\{y_1, \ldots, y_c\}$ el conjunto de etiquetas de clase.

- **Paso 1:** Si todos los ejemplos de S_t pertenecen a la misma clase y_i, se declara al nodo t como nodo hoja y se le etiqueta con y_i.
- **Paso 2:** Si S_t contiene ejemplos de más de una etiqueta de clase, se selecciona un atributo con el que particionar el conjunto de ejemplos S_t en subconjuntos más pequeños.

  Se crea un nodo hijo para cada valor del atributo seleccionado y se particiona el conjunto S_t atendiendo a estos valores.

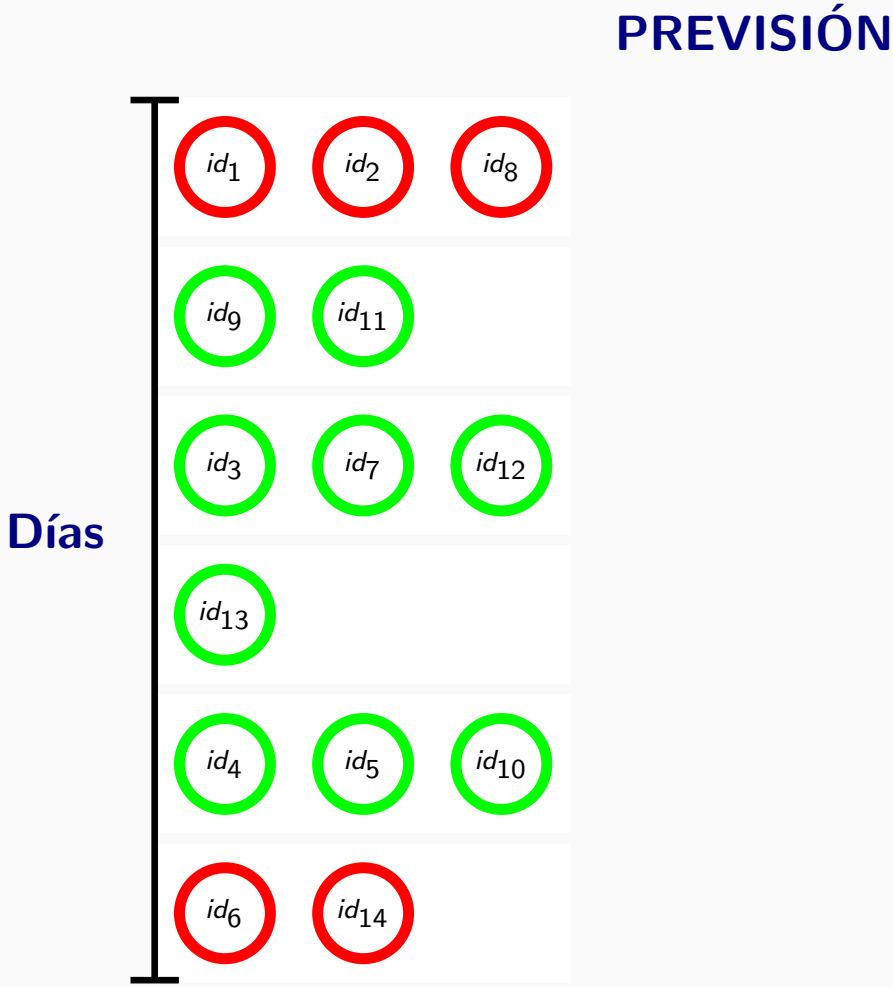
  A continuación se aplica recursivamente los pasos anteriores a cada nodo hijo.

Algoritmo de Hunt

Ejemplo (i)

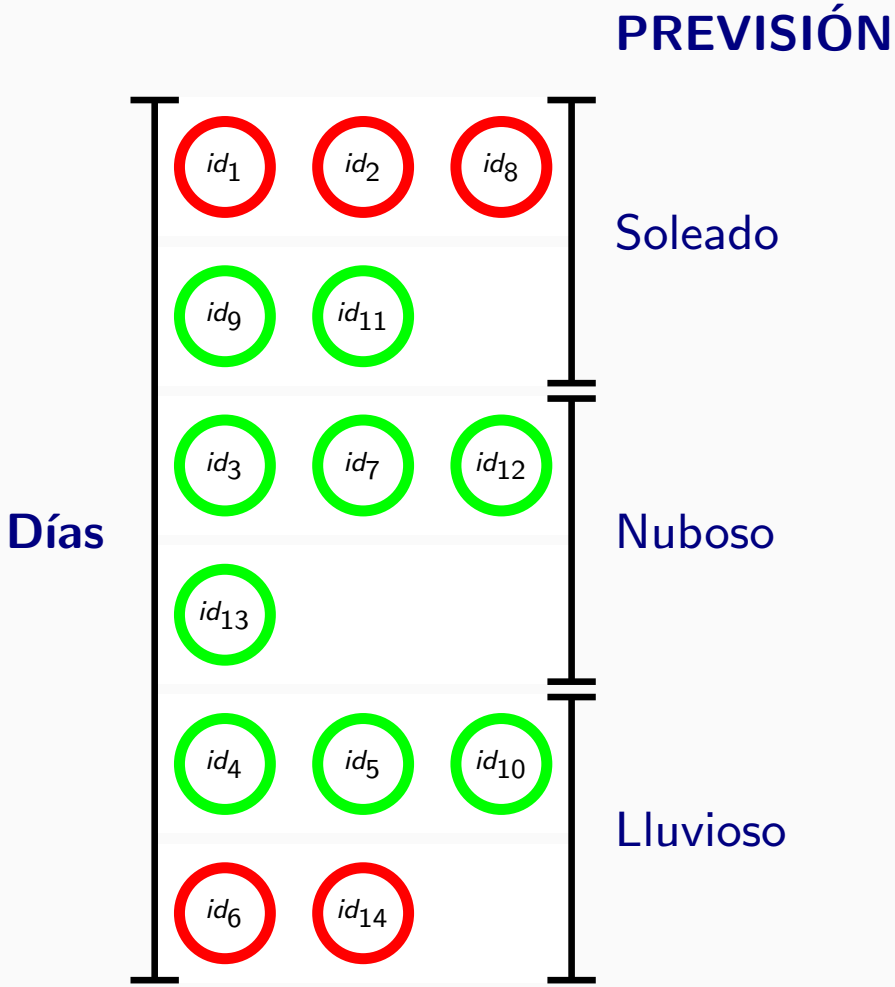
Algoritmo de Hunt

Ejemplo (i)



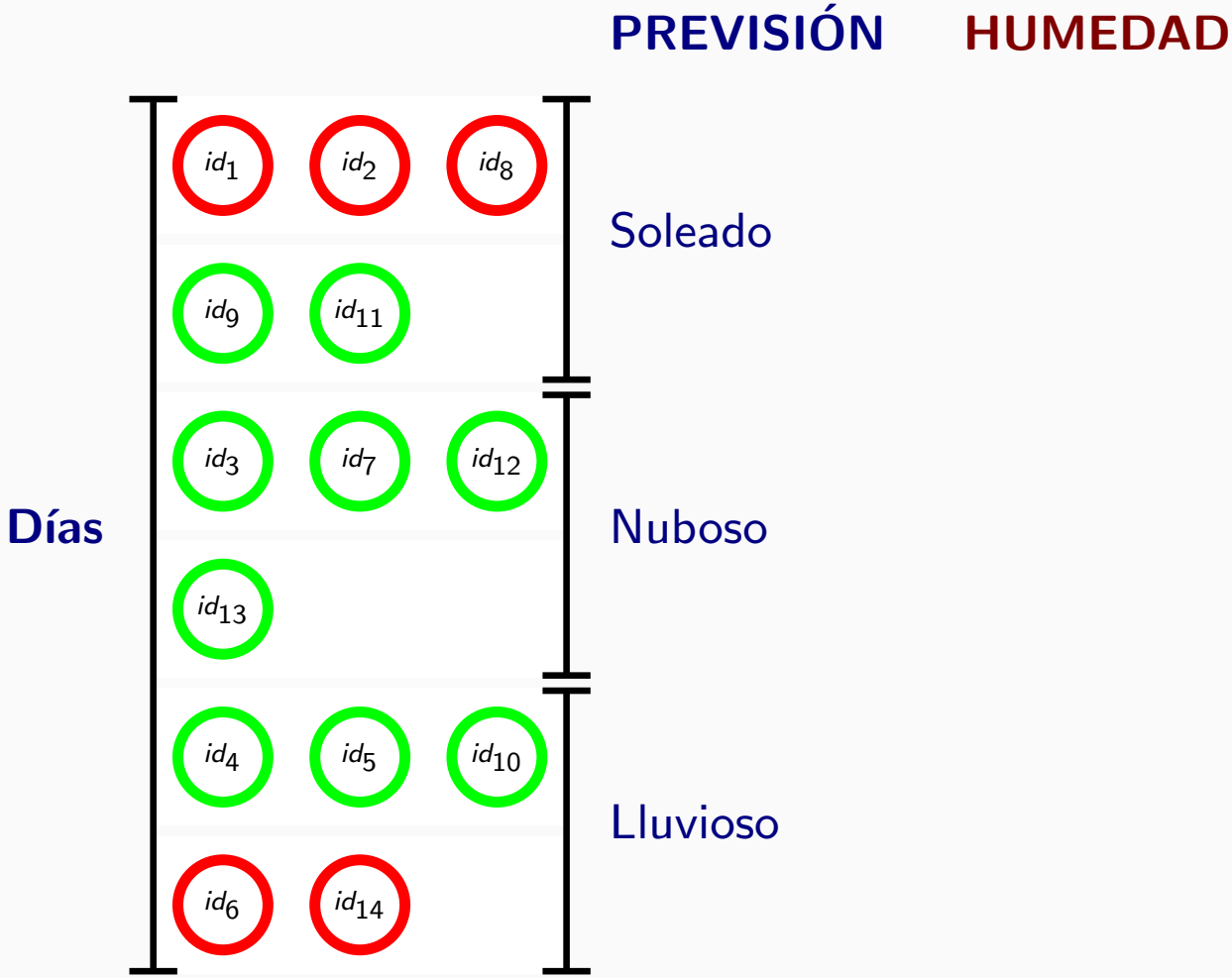
Algoritmo de Hunt

Ejemplo (i)



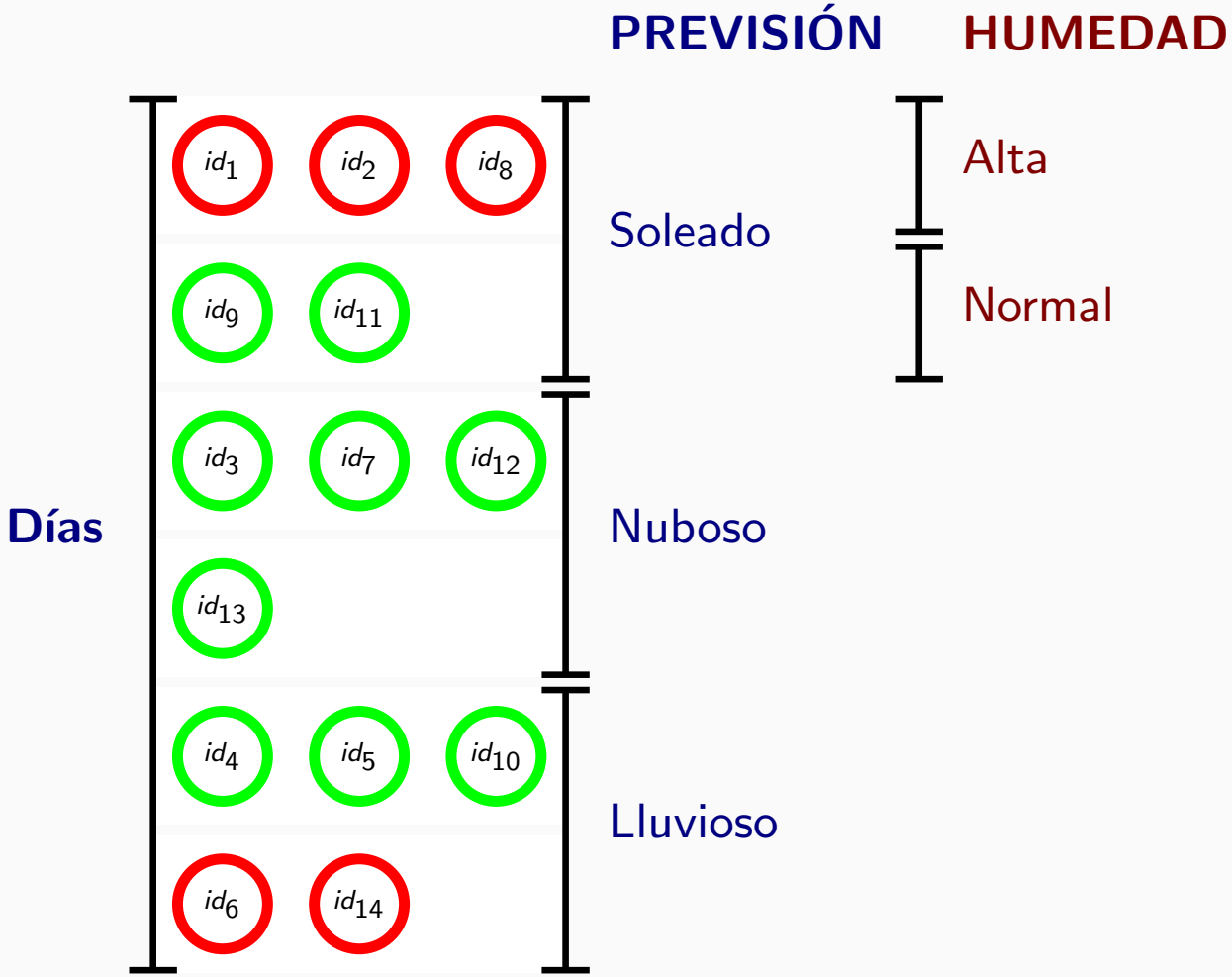
Algoritmo de Hunt

Ejemplo (i)



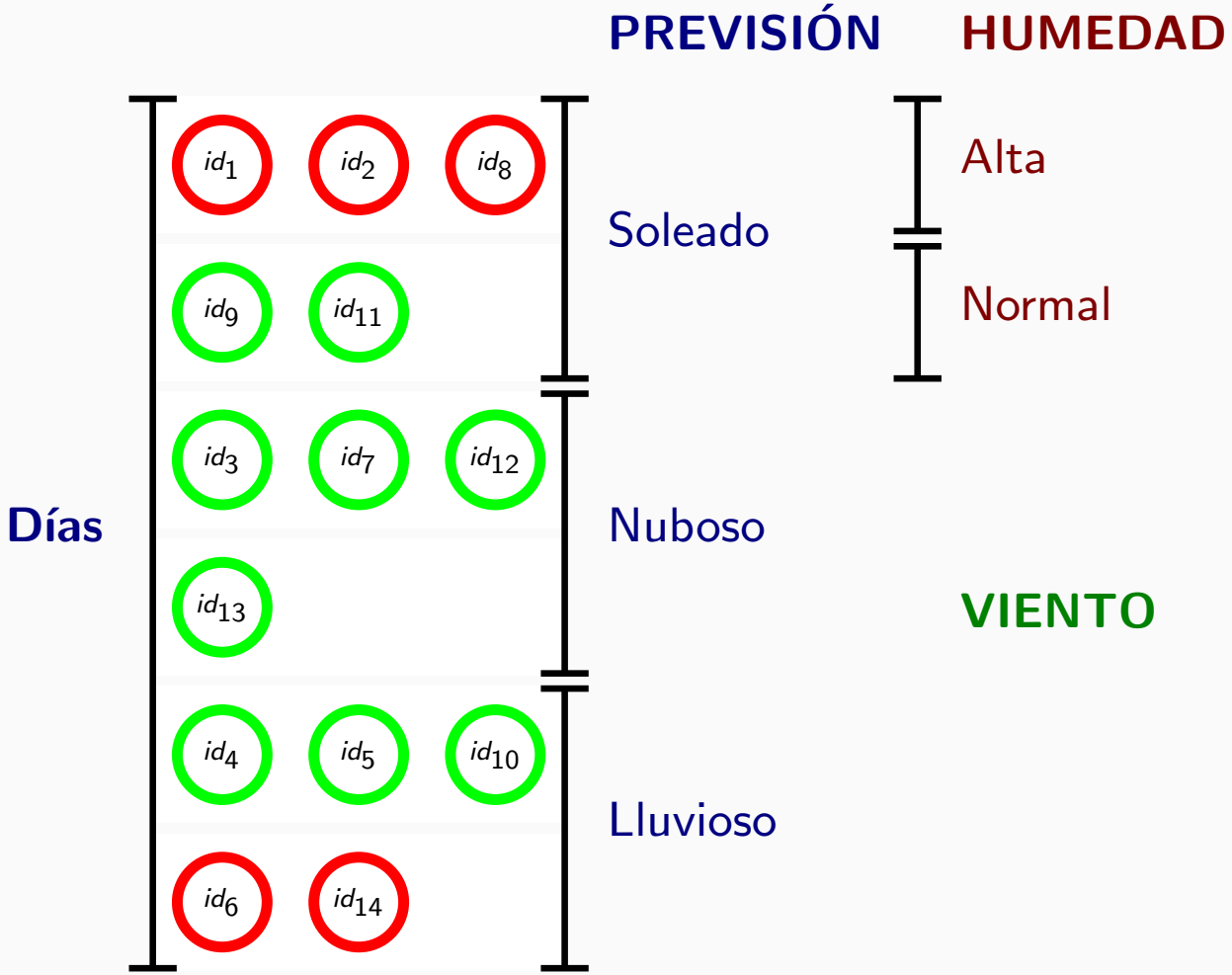
Algoritmo de Hunt

Ejemplo (i)



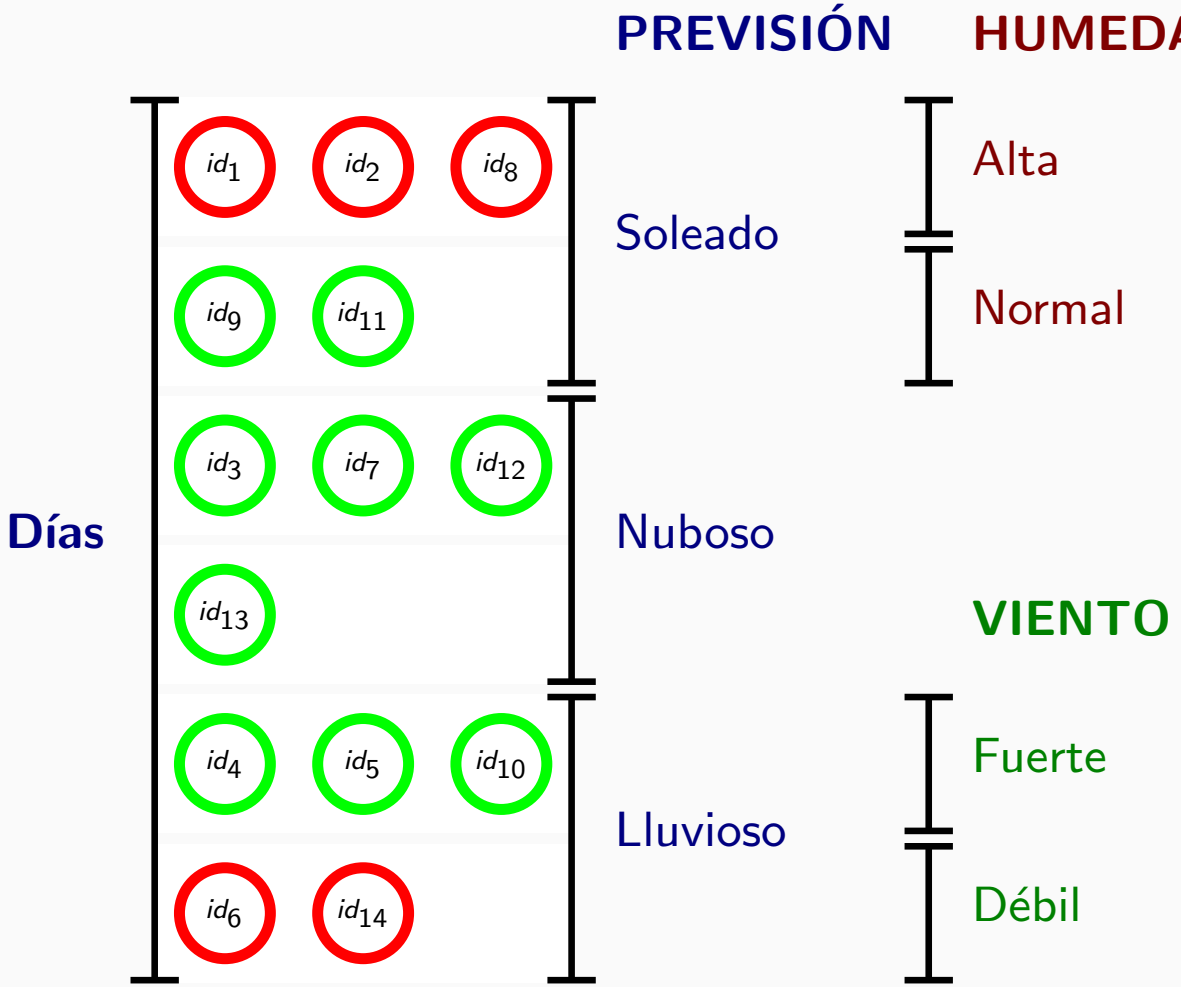
Algoritmo de Hunt

Ejemplo (i)



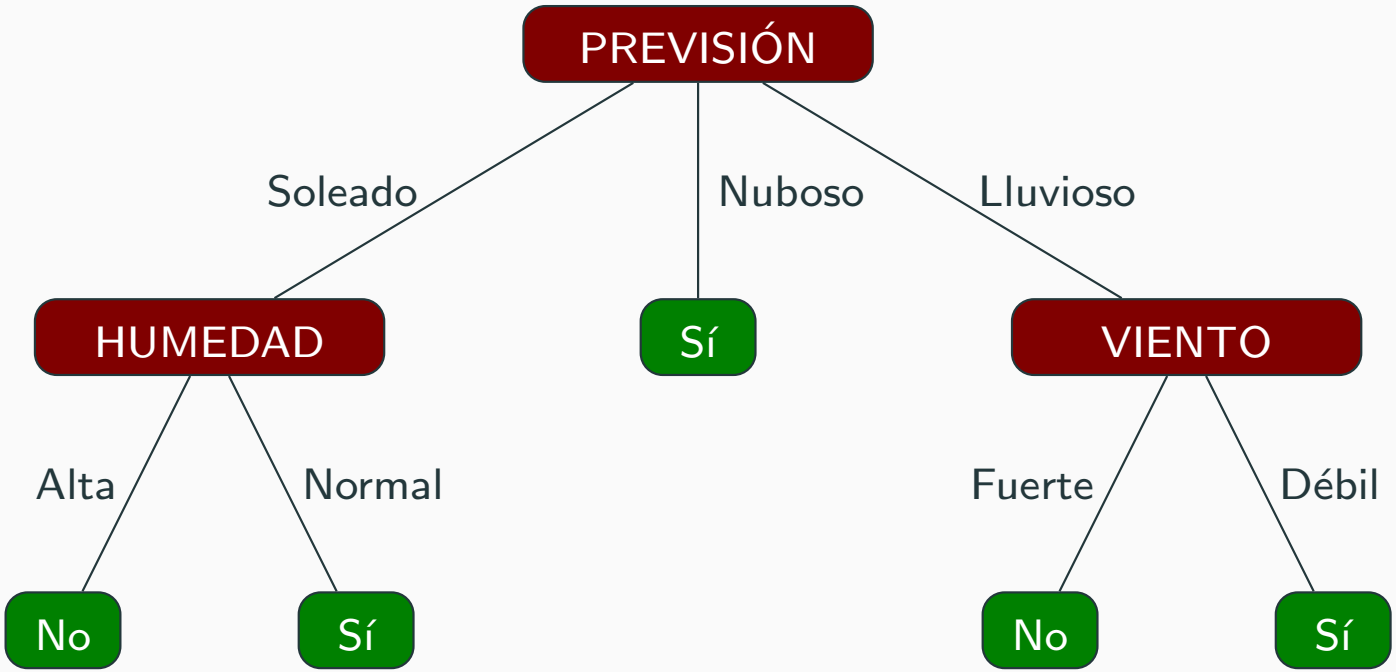
Algoritmo de Hunt

Ejemplo (i)



Algoritmo de Hunt

Ejemplo (ii)



Algoritmo de Hunt

Consideraciones

- Si para alguno de los nodos hijos creados en el paso 2 no existen ejemplos, el nodo se convierte en un nodo hoja con valor igual al de la clase más frecuente.

- Si todos los ejemplos de alguno de los subconjuntos S_t tienen los mismos valores de los atributos, pero distinto valor de la variable clase, el nodo se convierte en un nodo hoja con valor igual de de la clase más frecuente.

Algoritmo de Hunt

Cuestiones de diseño

- **¿Qué atributo escoger para dividir el conjunto de ejemplos?**. De todos los atributos disponibles, debe escogerse, en cada paso, uno con el que dividir el conjunto de ejemplos. En la literatura especializada se han propuesto diferentes criterios para seleccionar este atributo. Varios de estos criterios se basan en el grado de impureza de un conjunto.

- **¿Cuándo finalizar el proceso de división?** Dos condiciones de parada simples son parar cuando los ejemplos del correspodiente subconjunto tengan el mismo valor de la variable clase, o parar cuando no existan atributos con los que dividir este subconjunto. No obstante, existen otros criterios con los que obtener árboles simples con un alto poder de clasificación.

Algoritmo de Hunt

Grado de impureza de un conjunto

Sea $p(i)$ la fracción de ejemplos que pertenecen a la clase y_i en el conjunto S y c el número de clases. En el cálculo de la entropía se asume que $\log_2(0) = 0$.

- **Entropía:**

$$E(S) = -\sum_{i=1}^{c} p(i) \cdot \log_2 p(i)$$

- **Gini:**

$$G(S) = 1 - \sum_{i=1}^{c} [p(i)]^2$$

- **Error de clasificación:**

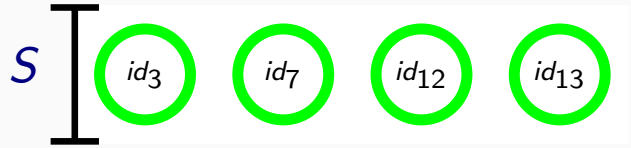
$$EC(S) = 1 - \max_{1 \leq i \leq n} \{p(i)\}$$

Grado de impureza de un conjunto. Ejemplo (i)

$S = \{id_1, id_2, \ldots, id_{14}\}$

y_i		$p(i)$	$\log_2(p(i))$	$E(S)$	$G(S)$	$EC(S)$
Sí	9	$9/14 = 0.642$	-0.637			
No	5	$5/14 = 0.358$	-1.485			
				0.940	0.459	0.358

Grado de impureza de un conjunto. Ejemplo (ii)



$S = \{id_3, id_7, id_{12}, id_{13}\}$

y_i		$p(i)$	$\log_2(p(i))$	$E(S)$	$G(S)$	$EC(S)$
Sí	4	$4/4 = 1.0$	0.0			
No	0	$0/4 = 0.0$	0.0			
				0.0	0.0	0.0

Grado de impureza. Ejemplo (ii)

$S_1 = \{id_1, id_2, id_8, id_9, id_{11}\}$

y_i		$p(i)$	$\log_2(p(i))$	$E(S_1)$	$G(S_1)$	$EC(S_1)$
Sí	2	$2/5 = 0.400$	-1.321			
No	3	$3/5 = 0.600$	-0.736			
				0.970	0.480	0.400

$S_2 = \{id_3, id_7, id_{12}, id_{13}\}$

y_i		$p(i)$	$\log_2(p(i))$	$E(S_2)$	$G(S_2)$	$EC(S_2)$
Sí	4	$4/4 = 1.000$	0.000			
No	0	$0/4 = 0.000$	0.000			
				0.000	0.000	0.000

$S_3 = \{id_4, id_5, id_6, id_{10}, id_{14}\}$

y_i		$p(i)$	$\log_2(p(i))$	$E(S_j)$	$G(S_j)$	$EC(S_j)$
Sí	3	$3/5 = 0.600$	-0.736			
No	2	$2/5 = 0.400$	-1.321			
				0.970	0.480	0.400

Algoritmo de Hunt

Qué atributo seleccionar para la división

- Para seleccionar el atributo con el que se dividirá el conjunto de ejemplos se compara el grado de impureza del conjunto (nodo padre) con el grado de impureza de los subconjuntos (nodos hijos) que se obtienen tras la división.
- La bondad de una partición se evalúa usando el concepto de ganancia que se define como la reducción promedio en el grado de impureza del conjunto.
- Se selecciona el atributo con el que se obtiene la mayor ganancia tras la división.
- Si se emplea como medida de impureza la entropía se habla de ganancia de información.

Algoritmo de Hunt

Ganancia

- Sea A un atributo con k valores distintos que particiona el conjunto de ejemplos S en los subconjuntos de ejemplos $S_1, \ldots, S_k$.
- Sean N y $N(S_j)$, $j = 1, \ldots, k$, respectivamente, los tamaños de los conjuntos S y S_j, $j = 1, \ldots, k$.
- La ganancia que se obtiene al particionar el conjunto S considerando el atributo A se define como

$$\Delta(S, A) = I(S) - \sum_{j=1}^{k} \frac{N(S_j)}{N} \cdot I(S_j)$$

donde $I(\cdot)$ es la medida de impureza considerada.

Ganancia. Ejemplo (i)

- Si se considera el atributo $A =$ Previsión, el conjunto $S = \{id_1, id_2 \ldots, id_{14}\}$ se particiona en los subconjuntos S_1, S_2 y S_3 anteriores.
- La ganancia obtenida usando como medida de impureza la entropía es

$$\begin{aligned}
\Delta(S, A) &= E(S) - \sum_{j=1}^{k} \frac{N(S_j)}{N} \cdot E(S_j) \\
&= 0.940 - \left(\frac{5}{14} \cdot 0.970 + \frac{4}{14} \cdot 0.000 + \frac{5}{14} \cdot 0.970 \right) \\
&= 0.247
\end{aligned}$$

Ganancia. Ejemplo (ii)

- Si A = Humedad, el conjunto $S = \{id_1, id_2 \ldots, id_{14}\}$ se particiona en los subconjuntos $S_1 = \{id_1, id_2, id_3, id_4, id_8, id_{12}, id_{14}\}$ (Humedad = Alta) y $S_2 = \{id_5, id_6, id_7, id_9, id_{10}, id_{11}, id_{13}\}$ (Humedad = Normal).
- La ganancia obtenida usando como medida de impureza la entropía es

$$\begin{aligned}
\Delta(S, A) &= E(S) - \sum_{j=1}^{k} \frac{N(S_j)}{N} \cdot E(S_j) \\
&= 0.940 - \left(\frac{7}{14} \cdot 0.985 + \frac{7}{14} \cdot 0.592 \right) \\
&= 0.151
\end{aligned}$$

Ganancia. Ejemplo (iii)

- Si A = Viento, el conjunto $S = \{id_1, id_2 \ldots, id_{14}\}$ se particiona en los subconjuntos $S_1 = \{id_2, id_6, id_7, id_{11}, id_{12}, id_{14}\}$ (Viento = Fuerte) y $S_2 = \{id_1, id_3, id_4, id_5, id_8, id_9, id_{10}, id_{13}\}$ (Viento = Débil).
- La ganancia obtenida usando como medida de impureza la entropía es

$$\begin{aligned}
\Delta(S, A) &= E(S) - \sum_{j=1}^{k} \frac{N(S_j)}{N} \cdot E(S_j) \\
&= 0.940 - \left(\frac{8}{14} \cdot 0.811 + \frac{6}{14} \cdot 1.000 \right) \\
&= 0.048
\end{aligned}$$

Ganancia. Ejemplo (iv)

- Si $A =$ Temperatura, el conjunto $S = \{id_1, id_2 \ldots, id_{14}\}$ se particiona en los subconjuntos $S_1 = \{id_1, id_2, id_3, id_9, id_{13}\}$ (Temperatura = Alta), $S_2 = \{id_4, id_8, id_{10}, id_{11}, id_{12}\}$ (Viento = Media) y $S_3 = \{id_5, id_6, id_7, id_{14}\}$ (Temperatura = Fría).
- La ganancia obtenida usando como medida de impureza la entropía es

$$\begin{aligned}
\Delta(S, A) &= E(S) - \sum_{j=1}^{k} \frac{N(S_j)}{N} \cdot E(S_j) \\
&= 0.940 - \left(\frac{5}{14} \cdot 0.970 + \frac{5}{14} \cdot 0.721 + \frac{4}{14} \cdot 1.000 \right) \\
&= 0.050
\end{aligned}$$

Algoritmo ID3

Descripción

- ID3 (Iterative Dichotomiser 3) es un algoritmo constructivo greedy para obtener árboles de decisión propuesto por Ross Quinlan [4].
- Se basa en el algoritmo CLS (Concept Learning Systems) de Hunt et al. [3].
- Coloca en el nodo raíz del árbol el atributo que por sí solo mejor clasifica el conjunto de entrenamiento. Para ello emplea la ganancia de información.
- A continuación crea un nodo hijo por cada posible valor del atributo y asigna los ejemplos de entrenamiento al nodo correspondiente.
- Los pasos anteriores se repiten con los ejemplos de entrenamiento asociados a cada nodo.

Observaciones

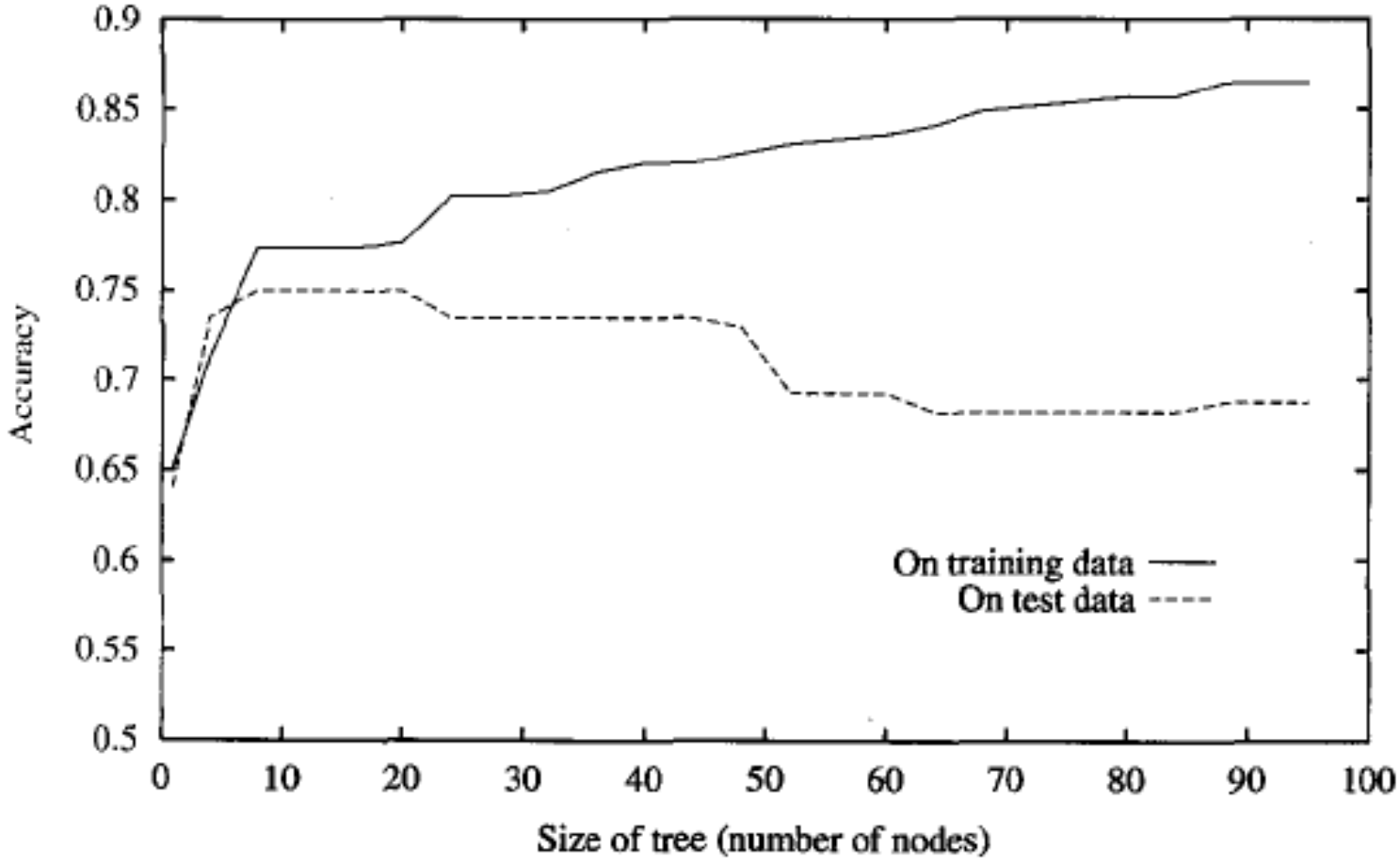
- El ser un algoritmo voraz (greedy) puede suministrar un árbol óptimo local en lugar de óptimo global.
- Su complejidad crece linealmente con el número de instancias de entrenamiento, pero exponencialmente con el número de atributos.
- Tiende a favorecer la elección de los atributos con mayor número de valores.
- Tiende a producir árboles que sobreajustan las instancias de entrenamiento.

Extensiones del algoritmo ID3

Sobreajuste (i)

- Se dice que un modelo sobreajusta el conjunto de entrenamiento cuando tiene un buen rendimiento en este conjunto, pero un rendimiento pobre en el conjunto de validación.
- El sobreajuste es común cuando se trabaja con pocas instancias de entrenamiento o cuando en estas hay ruido (valores incorrectos).

Sobreajuste (ii)



Extensiones del algoritmo ID3

Propuestas para evitar el sobreajuste

- **Simplificar el árbol**
  - **Pre-poda**. Parar el crecimiento del árbol antes de que se cumpla el criterio de finalización. En la práctica es difícil estimar cuándo debe podarse.
  - **Post-poda.** Tras la obtención del árbol, se poda este reemplazando algunos subárboles por nodos hojas. El coste computacional es alto.
- **Modificar el criterio de elección de los atributos**. Emplear para la elección de los atributos criterios que penalicen a los atributos con muchos valores.
- **Restringir el número de ramas.** Algunos algoritmos como CART [1] producen divisiones binarias del conjunto de ejemplos cuando evalúan a los atributos.

Extensiones del algoritmo ID3

Propuestas para evitar el sobreajuste

- **Simplificar el árbol**
  - **Pre-poda**. Parar el crecimiento del árbol antes de que se cumpla el criterio de finalización. En la práctica es difícil estimar cuándo debe podarse.
  - **Post-poda.** Tras la obtención del árbol, se poda este reemplazando algunos subárboles por nodos hojas. El coste computacional es alto.
- **Modificar el criterio de elección de los atributos**. Emplear para la elección de los atributos criterios que penalicen a los atributos con muchos valores.
- **Restringir el número de ramas.** Algunos algoritmos como CART [1] producen divisiones binarias del conjunto de ejemplos cuando evalúan a los atributos.

Extensiones del algoritmo ID3

Propuestas para evitar el sobreajuste

- **Simplificar el árbol**
  - **Pre-poda**. Parar el crecimiento del árbol antes de que se cumpla el criterio de finalización. En la práctica es difícil estimar cuándo debe podarse.
  - **Post-poda.** Tras la obtención del árbol, se poda este reemplazando algunos subárboles por nodos hojas. El coste computacional es alto.
- **Modificar el criterio de elección de los atributos**. Emplear para la elección de los atributos criterios que penalicen a los atributos con muchos valores.
- **Restringir el número de ramas.** Algunos algoritmos como CART [1] producen divisiones binarias del conjunto de ejemplos cuando evalúan a los atributos.

Propuestas para evitar el sobreajuste. Ejemplos (i)

- **Simplificar el árbol**

  - **Pre-poda**. Estimar si es probable que se produzca una mejora si se expande el nodo actual (Quinlan [4] aplicó el contraste de la χ^2 para este fin). Si no es probable, el nodo no se expande.

  - **Post-poda.** Dado un árbol de decisión, se poda un subárbol (convirtiéndolo en un nodo hoja con valor de clase igual a la clase más frecuente en el correspondiente nodo) si su poda produce un árbol con mejor rendimiento que el inicial. El proceso de poda comienza desde los nodos más profundos del árbol escogiendo, a igual nivel, el que produzca un mayor rendimiento.

Propuestas para evitar el sobreajuste. Ejemplos (i)

- **Simplificar el árbol**
  - **Pre-poda**. Estimar si es probable que se produzca una mejora si se expande el nodo actual (Quinlan [4] aplicó el contraste de la χ^2 para este fin). Si no es probable, el nodo no se expande.
  - **Post-poda.** Dado un árbol de decisión, se poda un subárbol (convirtiéndolo en un nodo hoja con valor de clase igual a la clase más frecuente en el correspondiente nodo) si su poda produce un árbol con mejor rendimiento que el inicial. El proceso de poda comienza desde los nodos más profundos del árbol escogiendo, a igual nivel, el que produzca un mayor rendimiento.

Propuestas para evitar el sobreajuste. Ejemplos (ii)

- **Modificar el criterio de elección de los atributos**
  - El ratio de ganancia, definido como

$$Ratio(S, A) = \frac{\Delta(S, A)}{Split(S, A)},$$

con

$$\Delta(S, A) = E(S) - \sum_{j=1}^{k} \frac{N(S_j)}{N} \cdot E(S_j)$$

y

$$Split(S, A) = -\sum_{j=1}^{k} \frac{N(S_j)}{N} \cdot \log_2 \left(\frac{N(S_j)}{N} \right),$$

penaliza los atributos con muchos valores.

Extensiones del algoritmos ID3

Cómo trabajar con atributos continuos

- Los atributos continuos se pueden discretizar dividiendo su rango en subintervalos.
- Una de las alternativas más usadas para discretizar un atributo continuo A identifica un umbral t y divide el rango en los subintervalos $A \leq t$ y $A > t$.
- El umbral t se selecciona como aquel que produce el mayor incremento en la medida de bondad de la división usada por el algoritmo.

Algoritmo C4.5

Descripción

- Algoritmo de inducción propuesto por Ross Quinlan [5] que usa el ratio de ganancia para seleccionar el atributo.
- Trabaja con atributos discretos y continuos. Los atributos continuos se evalúan considerando un umbral y dividiendo el conjunto de ejemplos en dos subconjuntos. Al primero pertenecen los ejemplos con valor del atributo menor o igual que el umbral y al segundo los ejemplos con valor mayor.
- Trabaja con valores perdidos en los atributos. A tal fin, los ejemplos con valores perdidos en algún atributo no se usan en el cálculo de la entropía o de la ganancia.
- Trabaja con atributos con costes diferentes.
- Poda los árboles después de su creación.

Algoritmo C4.5

Descripción

- Algoritmo de inducción propuesto por Ross Quinlan [5] que usa el ratio de ganancia para seleccionar el atributo.
- Trabaja con atributos discretos y continuos. Los atributos continuos se evalúan considerando un umbral y dividiendo el conjunto de ejemplos en dos subconjuntos. Al primero pertenecen los ejemplos con valor del atributo menor o igual que el umbral y al segundo los ejemplos con valor mayor.
- Trabaja con valores perdidos en los atributos. A tal fin, los ejemplos con valores perdidos en algún atributo no se usan en el cálculo de la entropía o de la ganancia.
- Trabaja con atributos con costes diferentes.
- Poda los árboles después de su creación.

Algoritmo C4.5

Descripción

- Algoritmo de inducción propuesto por Ross Quinlan [5] que usa el ratio de ganancia para seleccionar el atributo.
- Trabaja con atributos discretos y continuos. Los atributos continuos se evalúan considerando un umbral y dividiendo el conjunto de ejemplos en dos subconjuntos. Al primero pertenecen los ejemplos con valor del atributo menor o igual que el umbral y al segundo los ejemplos con valor mayor.
- Trabaja con valores perdidos en los atributos. A tal fin, los ejemplos con valores perdidos en algún atributo no se usan en el cálculo de la entropía o de la ganancia.
- Trabaja con atributos con costes diferentes.
- Poda los árboles después de su creación.

Algoritmo C4.5

Descripción

- Algoritmo de inducción propuesto por Ross Quinlan [5] que usa el ratio de ganancia para seleccionar el atributo.
- Trabaja con atributos discretos y continuos. Los atributos continuos se evalúan considerando un umbral y dividiendo el conjunto de ejemplos en dos subconjuntos. Al primero pertenecen los ejemplos con valor del atributo menor o igual que el umbral y al segundo los ejemplos con valor mayor.
- Trabaja con valores perdidos en los atributos. A tal fin, los ejemplos con valores perdidos en algún atributo no se usan en el cálculo de la entropía o de la ganancia.
- Trabaja con atributos con costes diferentes.
- Poda los árboles después de su creación.

Algoritmo C4.5

Descripción

- Algoritmo de inducción propuesto por Ross Quinlan [5] que usa el ratio de ganancia para seleccionar el atributo.
- Trabaja con atributos discretos y continuos. Los atributos continuos se evalúan considerando un umbral y dividiendo el conjunto de ejemplos en dos subconjuntos. Al primero pertenecen los ejemplos con valor del atributo menor o igual que el umbral y al segundo los ejemplos con valor mayor.
- Trabaja con valores perdidos en los atributos. A tal fin, los ejemplos con valores perdidos en algún atributo no se usan en el cálculo de la entropía o de la ganancia.
- Trabaja con atributos con costes diferentes.
- Poda los árboles después de su creación.

Atributos continuos

- Sea A un atributo continuo y $S = \{s_1, s_2, \ldots, s_n\}$ el conjunto de ejemplos que se desea particionar.
- Supóngase que el conjunto S está ordenado de tal forma que $s_i \leq s_{i+1}$, para todo $i \in \{1, 2, \ldots, n-1\}$.
- Los umbrales potenciales que deben considerarse son:

$$t_i = \frac{s_i + s_{i+1}}{2}, \qquad i = 1, 2, \ldots, n-1$$

- De los anteriores umbrales potenciales se seleccionará aquel con el que se obtenga el mayor ratio de ganancia.
- Fayyad e Irani [2] probaron que solo es necesario considerar los umbrales en los que se produzca un cambio en el valor de la variable clase.

Atributos continuos

- Sea A un atributo continuo y $S = \{s_1, s_2, \ldots, s_n\}$ el conjunto de ejemplos que se desea particionar.
- Supóngase que el conjunto S está ordenado de tal forma que $s_i \leq s_{i+1}$, para todo $i \in \{1, 2, \ldots, n-1\}$.
- Los umbrales potenciales que deben considerarse son:

$$t_i = \frac{s_i + s_{i+1}}{2}, \qquad i = 1, 2, \ldots, n-1$$

- De los anteriores umbrales potenciales se seleccionará aquel con el que se obtenga el mayor ratio de ganancia.
- Fayyad e Irani [2] probaron que solo es necesario considerar los umbrales en los que se produzca un cambio en el valor de la variable clase.

Atributos continuos

- Sea A un atributo continuo y $S = \{s_1, s_2, \ldots, s_n\}$ el conjunto de ejemplos que se desea particionar.
- Supóngase que el conjunto S está ordenado de tal forma que $s_i \leq s_{i+1}$, para todo $i \in \{1, 2, \ldots, n-1\}$.
- Los umbrales potenciales que deben considerarse son:

$$t_i = \frac{s_i + s_{i+1}}{2}, \qquad i = 1, 2, \ldots, n-1$$

- De los anteriores umbrales potenciales se seleccionará aquel con el que se obtenga el mayor ratio de ganancia.
- Fayyad e Irani [2] probaron que solo es necesario considerar los umbrales en los que se produzca un cambio en el valor de la variable clase.

Atributos continuos

- Sea A un atributo continuo y $S = \{s_1, s_2, \ldots, s_n\}$ el conjunto de ejemplos que se desea particionar.
- Supóngase que el conjunto S está ordenado de tal forma que $s_i \leq s_{i+1}$, para todo $i \in \{1, 2, \ldots, n-1\}$.
- Los umbrales potenciales que deben considerarse son:

$$t_i = \frac{s_i + s_{i+1}}{2}, \qquad i = 1, 2, \ldots, n-1$$

- De los anteriores umbrales potenciales se seleccionará aquel con el que se obtenga el mayor ratio de ganancia.
- Fayyad e Irani [2] probaron que solo es necesario considerar los umbrales en los que se produzca un cambio en el valor de la variable clase.

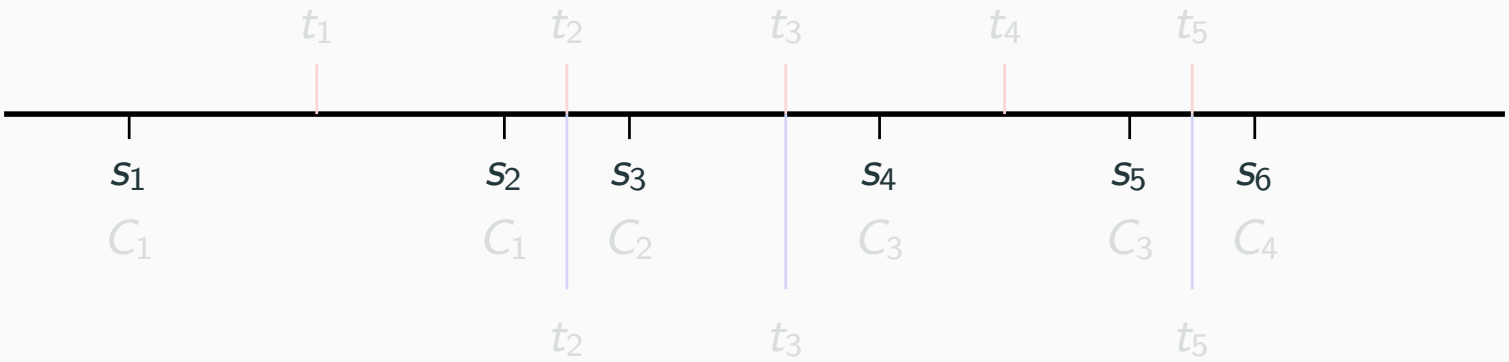
Atributos continuos

- Sea A un atributo continuo y $S = \{s_1, s_2, \ldots, s_n\}$ el conjunto de ejemplos que se desea particionar.
- Supóngase que el conjunto S está ordenado de tal forma que $s_i \leq s_{i+1}$, para todo $i \in \{1, 2, \ldots, n-1\}$.
- Los umbrales potenciales que deben considerarse son:

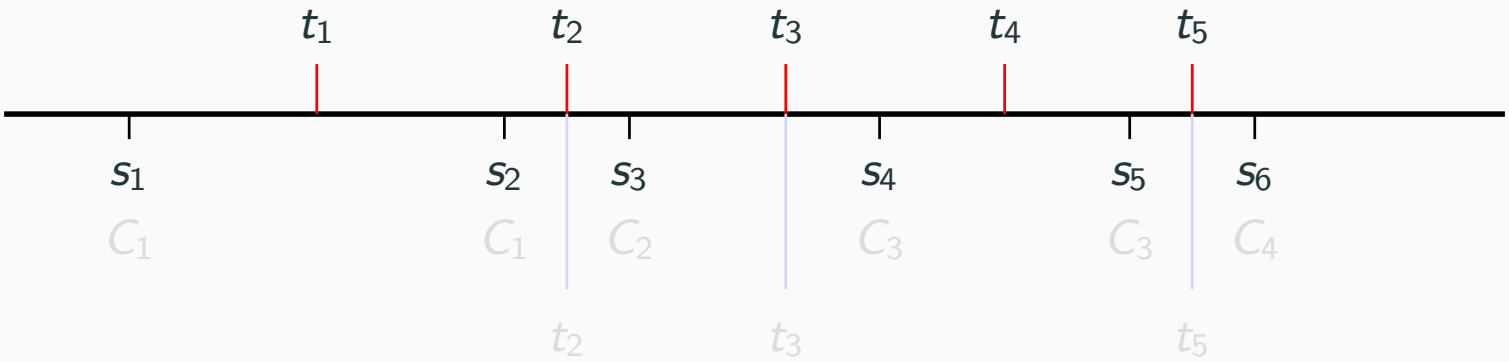
$$t_i = \frac{s_i + s_{i+1}}{2}, \qquad i = 1, 2, \ldots, n-1$$

- De los anteriores umbrales potenciales se seleccionará aquel con el que se obtenga el mayor ratio de ganancia.
- Fayyad e Irani [2] probaron que solo es necesario considerar los umbrales en los que se produzca un cambio en el valor de la variable clase.

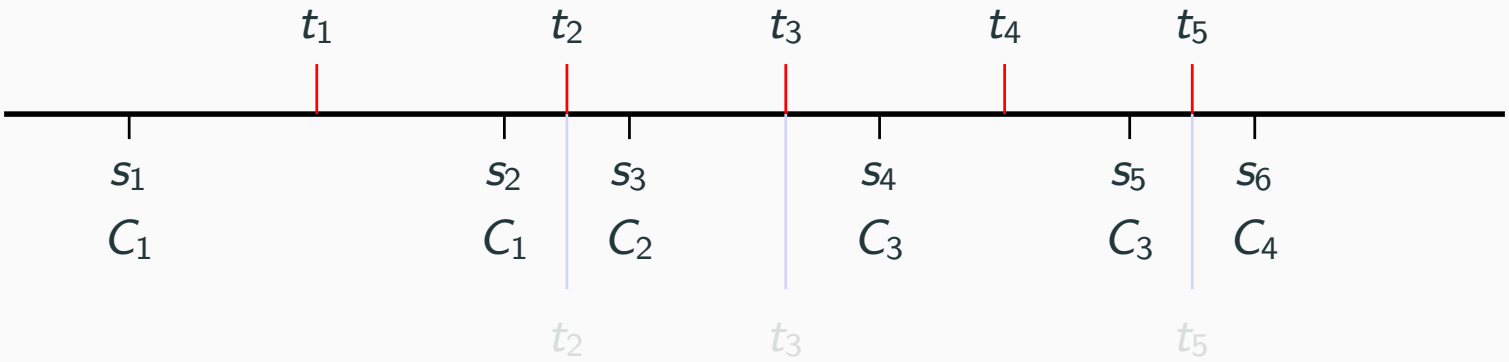
Atributos continuos (ii)



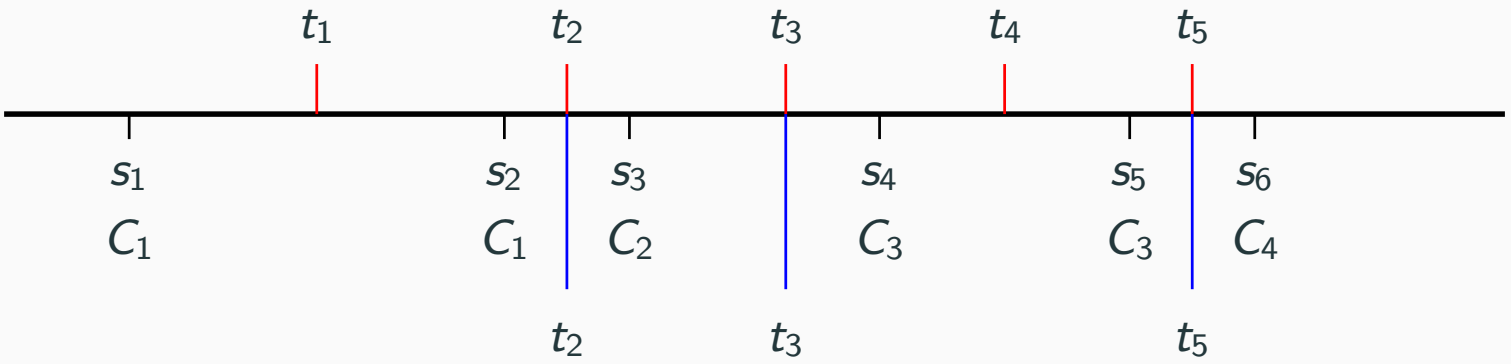
Atributos continuos (ii)



Atributos continuos (ii)



Atributos continuos (ii)



Algoritmo C4.5

Poda

- Convertir el árbol de decisión en un conjunto de reglas R.
- Sea *Mejor_Error* el error al clasificar con el conjunto de reglas R. Inicialmente coincide con el error de clasificación del árbol de decisión.
- Para cada regla $r \in R$
  - Para cada antecedente j de la regla r
    - Sea *Error* el error que se produce al clasificar eliminando j de la regla r.
    - Si $Error \leq Mejor\_Error$, entonces $Mejor\_Error = Error$ y se elimina j de r.
  - Si no hay antecedentes en r, eliminar r de R.

Poda

- Convertir el árbol de decisión en un conjunto de reglas R.
- Sea *Mejor_Error* el error al clasificar con el conjunto de reglas R. Inicialmente coincide con el error de clasificación del árbol de decisión.
- Para cada regla $r \in R$
  - Para cada antecedente j de la regla r
    - Sea *Error* el error que se produce al clasificar eliminando j de la regla r.
    - Si $Error \leq Mejor\_Error$, entonces $Mejor\_Error = Error$ y se elimina j de r.
  - Si no hay antecedentes en r, eliminar r de R.

Poda

- Convertir el árbol de decisión en un conjunto de reglas R.
- Sea *Mejor_Error* el error al clasificar con el conjunto de reglas R. Inicialmente coincide con el error de clasificación del árbol de decisión.
- Para cada regla $r \in R$
  - Para cada antecedente j de la regla r
    - Sea *Error* el error que se produce al clasificar eliminando j de la regla r.
    - Si $Error \leq Mejor\_Error$, entonces $Mejor\_Error = Error$ y se elimina j de r.
  - Si no hay antecedentes en r, eliminar r de R.

BIBLIOGRAFÍA

Bibliografía I

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J.

Classification and Regression Trees.

Chapman and Hall, 1984.

Fayyad, U. M., and Irani, K. B.

On the handling of continuous-valued attributes in decision tree generation.

Machine Learning 8 (1992), 87–102.

Hunt, E. B., Marin, J., and Stone, P. J.

Experiments in induction.

Academic Press, 1966.

Bibliografía II


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


Quinlan, J. R.
C4.5 Programs for Machine Learning.
Morgan Kaufmann, 1993.


Tan, P.-N., Steinbach, M., and Kumar, V.
Introduction to Data Mining.
Addison-Wesley, 2006.

Licencia

Esta obra está bajo una licencia de Creative Commons.
Reconocimiento - No comercial - Compartir igual