3.3. APRENDIZAJE DE ÁRBOLES DE DECISIÓN: ID3 Y C4.5.

- **Generalidades**

El aprendizaje de árboles de decisión está englobado como una metodología del aprendizaje supervisado. La representación que se utiliza para las descripciones del concepto adquirido es el árbol de decisión, que consiste en una representación del conocimiento relativamente simple por lo que los procedimientos utilizados en su aprendizaje son más sencillos que los de sistemas que utilizan lenguajes de representación más potentes, como redes semánticas, representaciones en lógica de primer orden etc.. Como contrapartida la potencia expresiva de los árboles de decisión es también menor que la de esos otros sistemas. El aprendizaje de árboles de decisión suele ser más robusto frente al ruido y conceptualmente sencillo, aunque los sistemas que han resultado del perfeccionamiento y de la evolución de los más antiguos se complican con los procesos que incorporan para ganar fiabilidad.

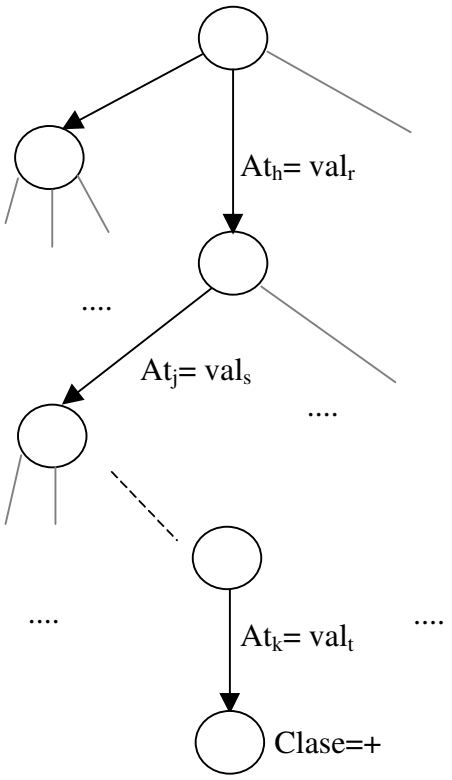


Figura 26: Árbol de decisión

La mayoría de los sistemas de aprendizaje de árboles suelen ser no incrementales, pero existe alguna excepción, como se verá más adelante [UTGOFF 89].

El primer sistema que construía árboles de decisión fue CLS de Hunt, desarrollado en 1959 y depurado a lo largo de los años sesenta. CLS es un sistema desarrollado por psicólogos como un modelo del proceso cognitivo de formación de conceptos sencillos. Su contribución fundamental fue la propia metodología pero no resultaba computacionalmente eficiente debido al método que empleaba en la extensión de los nodos. Se guiaba por una estrategia similar al *minimax* con una función que integraba diferentes costes.

En 1979 Quinlan desarrolla el sistema ID3 [QUINLAN 79], que él denominaría simplemente herramienta porque la consideraba experimental. Conceptualmente es fiel a la metodología de CLS pero le aventaja en el método de expansión de los nodos basado en una función que utiliza la medida de la información de Shannon. La versión definitiva, presentada por su autor Quinlan como un sistema de aprendizaje, es el sistema C4.5 que expone con cierto detalle en la obra *C4.5: Programs for Machine Learning* [QUINLAN 93], incluye el código fuente viene acompañado de un disquete con el software. La última versión de este software está disponible en Internet (Universidad de Sydney). La evolución -comercial- de ese sistema es el otro denominado C5 del mismo autor, del que se puede obtener una versión de demostración restringida en cuanto a capacidades ; por ejemplo, el número máximo de ejemplos de entrenamiento.

- **Representación de un árbol de decisión**

Un árbol de decisión puede interpretarse esencialmente como una serie de *reglas* compactadas para su representación en forma de árbol. Dado un conjunto de ejemplos, estructurados como vectores de pares ordenados atributo-valor, de acuerdo con el formato general en el aprendizaje inductivo a partir de ejemplos, el concepto que estos sistemas adquieren durante el proceso de aprendizaje consiste en un árbol con el esquema que aparece en la **figura 26**. Cada eje están etiquetado con un par atributo-valor y las hojas con una clase, de forma que la trayectoria que determinan desde la raíz los pares de un ejemplo de entrenamiento alcanza una hoja etiquetada -normalmente- con la clase del ejemplo. La clasificación de un ejemplo nuevo del que se desconoce su clase se hace con la misma técnica, solamente que en ese caso al atributo clase, cuyo valor se desconoce, se le asigna de acuerdo con la etiqueta de la hoja a la que se accede con ese ejemplo.

- **Problemas apropiados para este tipo de aprendizaje**

Las características de los problemas apropiados para resolver mediante este aprendizaje dependen del sistema de aprendizaje específico utilizado, pero hay una serie de ellas generales y comunes a la mayoría. Pasamos describirlas:

- Que la representación de los ejemplos sea mediante vectores de pares atributo-valor, especialmente cuando los ***valores son disjuntos*** y en un ***número pequeño***. Los sistemas actuales están preparados para tratar atributos con valores continuos, valores desconocidos e incluso valores con una distribución de probabilidad.
- Que el atributo que hace el papel de la clase sea de ***tipo discreto*** y con un número *pequeño* de valores, sin embargo existen sistemas que adquieren como concepto aprendido funciones con valores continuos.
- Que las ***descripciones del concepto adquirido deban ser expresadas en forma normal disyuntiva***.
- Que posiblemente **existan errores de clasificación** en el conjunto de ejemplos de entrenamiento, así como ***errores en los valores de sus atributos***. Estos sistemas, por lo general, son robustos frente a los errores del tipo mencionado, incluso algunos manejan bien los valores desconocidos de los atributos.

3.3.1. ESTUDIO DEL SISTEMA C4.5.

C4.5 es la versión completada de la herramienta que Quinlan denominó ID3. Aprovechamos la historia de estos dos sistemas para presentarlos secuencialmente de manera que la complejidad aparezca en las explicaciones de modo paulatino. ID3, que es conceptualmente sencillo, ayuda a captar la metodología de este tipo de aprendizaje. C4.5 como culminación del proceso creativo, aunque no es definitivo como lo demuestran las sucesivas versiones que van apareciendo, es bastante más complejo; pero su base conceptual es la de su predecesor. De esta forma, estudiado ID3 , C4.5 es su continuación salvo algunas diferencias cuya explicación constituye el grueso del estudio de C4.5.

3.3.1.1. ID3 un sistema básico en el aprendizaje de árboles

- **El esquema principal del algoritmo**

En el proceso de construcción del árbol de decisión, además de etiquetar los arcos y las hojas como ya se indicó, se etiquetan los nodos con un subconjunto de ejemplos de entrenamiento. Las operaciones básicas del proceso esquematizado a muy bajo nivel serán:

- Todo nodo está etiquetado con un subconjunto de ejemplos de los de entrenamiento. La raíz estará siempre etiquetada con el conjunto total de ejemplos de entrenamiento.
- Todo nodo que no constituya una hoja se expande.
- El proceso se detendrá cuando se haya alcanzado hojas por todos los caminos.

El principio sobre el que se basa la expansión de los nodos del árbol que se trata de aprender o construir a partir del conjunto de ejemplos de entrenamiento es maquiavélico: *divide y vencerás* [QUINLAN 83].

Ya que cada nodo estará etiquetado por cierto subconjunto de ejemplos de entrenamiento, supongamos que en el desarrollo del árbol se alcanzó un nodo etiquetado por el conjunto E de ejemplos.

Si E tuviese todos los ejemplos de una misma clase entonces el nodo se constituiría en una hoja del árbol y esa clase en su etiqueta.

Por el contrario, si los ejemplos de E son de clases diferentes, mediante un proceso que Quinlan denomina ***test,*** y que más adelante detallaremos, se determina el atributo cuyos valores *mejor parten* al conjunto E (un atributo con sus valores induce una relación de equivalencia o partición sobre E) en el sentido de ajustarse lo más posible a la partición que el atributo clase determina también sobre E; es decir, que la partición que produce el *mejor* atributo sobre E será la que más se *parezca* a la partición que produce el atributo clase.

Determinado el *mejor* atributo, se generarán tantos hijos del nodo en estudio como valores tuviese el atributo. Cada uno de los nuevos arcos se etiquetará con un valor del atributo y los correspondientes nodos hijo con el subconjunto de ejemplos de E que posean ese valor del atributo.

Así, el proceso principal trata, en sucesivas iteraciones, de subdividir el conjunto de ejemplos hasta alcanzar subconjuntos con todos los elementos en una misma clase.

La elección del *mejor* atributo para expandir un nodo la efectúa ID3 interpretando que el *mejor* será aquel que más *información* proporcione sobre el atributo clase, o sea, que el conocimiento del valor del atributo en un ejemplo cualquiera de los que etiquetan el nodo permita predecir el valor de su clase con mayor seguridad.

La formalización del concepto de medida de la información, fue introducido por Shannon en 1948. A continuación desarrollamos los conceptos básicos de esta teoría de la medida de la información, necesarios para una compresión completa de ID3; además, actualmente sus ramificaciones salpican no sólo el aprendizaje automático sino todas las Ciencias de la Computación.

- **Los conceptos de medida de la información y de la entropía o incertidumbre de una fuente de memoria nula.**

Shannon definió una medida para la información basada en el concepto de probabilidad.

> **Definición 3.1.** Sea E un suceso que puede presentarse con probabilidad p(E). Cuando E tiene lugar, decimos que hemos recibido $I(E)= \log_a(1/p(E))$ unidades de información. Conforme ***a*** sea el número ***e***, el 10 o el 2 la unidad recibe, respectivamente, el nombre de ***nat***, ***hartley*** (en honor a Hartley que fue el primero en sugerir una medida para la información en 1928) o ***bit***.

Obsérvese que si se trabaja en base 2 y p(E)= 1/2 entonces I(E)= 1 bit; lo que permite definir el bit como la cantidad de información proporcionada por la especificación de uno de dos sucesos equiprobables.

El concepto de información se generaliza para un mecanismo generador de información denominado fuente de información de memoria nula.

Definición 3.2. Sea S= $\{s_1, s_2, \ldots s_n\}$ un alfabeto finito y fijo que se utilizará para transmitir la información proporcionada por un dispositivo generador denominado fuente. Si los símbolos, o elementos de S, son transmitidos sucesivamente de acuerdo a una distribución de probabilidad fija: $p(s_1)$, $p(s_2)$, ... , $p(s_n)$, entonces se denomina fuente de información de memoria nula. Por abuso del lenguaje se suele identificar la fuente con el propio alfabeto S.

En una fuente de información de memoria nula la emisión de un símbolo es independiente de los símbolos emitidos anteriormente; es decir, la distribución de probabilidad es fija, de ahí que se denomine de memoria nula.

Definición 3.3. Sea S= $\{s_1, s_2, \ldots s_n\}$ el alfabeto de una fuente de información de memoria nula. La cantidad media de información por símbolo del alfabeto S se denomina entropía de S, se representa por H(S), y tiene la siguiente expresión:

$$H(S)= \sum_{i=1}^{n} p(s_i) \log_2 (1/p(s_i))$$

Otra forma de interpretar la entropía de una fuente de información de memoria nula es apelando al concepto intuitivo de *incertidumbre*. H(S) puede interpretarse como el *valor medio de la incertidumbre* de un observador antes de conocer una salida de la fuente. Cualquiera de las dos interpretaciones puede utilizarse según lo aconseje la situación.

- **Algunas propiedades elementales de la entropía**

Algunas propiedades sencillas, y útiles en nuestro entorno, de la entropía de una fuente de información de memoria nula S son:

- $0 \leq H(S) \leq \log_2(\text{card}(S))$; es decir la entropía máxima de una fuente crece *lentamente* con el cardinal del alfabeto.
- $H(S) = \log_2(\text{card}(S))$, o sea, es máxima con un cierto alfabeto S, exactamente cuando los símbolos de S son equiprobables.
- $H(S) = 0$ exactamente cuando $p(s_j) = 1$ y $p(s_j) = 0$ $\forall j \neq i$ con $s_i, s_j \in S$.
- Sea ω la probabilidad de uno de los dos símbolos de un alfabeto binario S. En la **figura 27** se representa la curva de la variación de H(S) en función de ω. Téngase en cuenta que por definición se establece que $0 \log_2 0 = 0$

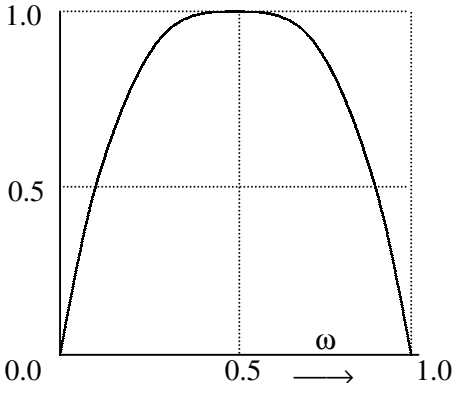


figura 27.

Ejemplo 3.4. Una imagen vale por mil palabras
Como una aplicación directa de la medida de la información vamos a desarrollar un sencillo

ejemplo muy intuitivo y que proporcionará una pequeña muestra de las posibilidades que brinda. Compararemos mediante la medida de Shannon, y en unas condiciones no del todo reales pero suficientes para un ámbito académico, la cantidad de información guardada por dos medios diferentes de transmisión de información. Consideraremos, por un lado, una sola imagen de televisión de 500 filas por 600 columnas con una escala de 10 grises -blanco y negro incluidos- y, por otro lado, un discurso de mil palabras, proporcionado por un locutor cuyo vocabulario consta de 10.000.

Aproximaremos las probabilidades que se utilizarán mediante frecuencias relativas.

El número de puntos por imagen es 500×600 y, considerando que cada punto puede tener uno de los 10 grises posibles, se tendrán $10^{500 \times 600} = 10^{300.000}$ imágenes posibles -no entramos en el significado de las imágenes. Entonces, la probabilidad de una imagen de televisión de las características citadas, y suponiendo que todas las imágenes posibles son igualmente probables, es $1/10^{300.000}$. Luego la información proporcionada por una de estas imágenes es:

$$I(\textit{imagen}) = \log_2(1/\text{prob.}(\textit{imagen})) = \log_2 10^{300.000} = 300.000 \log_2 10$$

Si consideramos que los comentarios de 1000 palabras efectuados por el locutor que posee un vocabulario de 10.000 palabras son igualmente probables -tampoco entramos en el significado-, la probabilidad de uno de estos comentarios es $1/10.000^{1000}$, por lo que la información guardada por un comentario de esas características es:

$$I(\textit{comentario}) = \log_2(1/\text{prob.}(\textit{comentario})) = \log_2 10.000^{1.000} = 1000 \log_2 10.000 = 4000 \log_2 10$$

Se tiene, pues, que $I(\textit{imagen}) = 75 \times I(\textit{comentario})$. Una de *esas* imágenes *equivale* a 75 de *esos* comentarios. Así pues, :*una imagen vale por 75.000 palabras*

Después de esta pequeña introducción a la teoría de la medida de la información, retomamos el estudio del ID3.

- **Los árboles de decisión y las fuentes de información de memoria nula**

Quinlan hace una interpretación intuitiva de los árboles de decisión por la que cada nodo del árbol puede interpretarse como una fuente de información de memoria nula con tantos símbolos de salida como clases o categorías existan. Cada nodo etiquetado con un subconjunto de ejemplos *guarda* una información media sobre las clase de esos ejemplos o, alternativamente, supone para un espectador una incertidumbre a la hora de predecir la clase de uno de esos ejemplos.

- **La elección del *mejor* atributo**

A partir de la idea anterior la elección del mejor atributo en un nodo N etiquetado con un conjunto de ejemplos E_i se decide mediante un **test** aplicable a cada atributo $\mathbf{X}$[1]**,** de los que no se utilizaron en la expansión de un nodo previo, de la siguiente manera, **fig. 3.4**:

[1] Quinlan habla de un test(X) originado por el atributo A. Aunque es más preciso ese lenguaje, aquí hablaremos directamente del test(A); es decir, se identifica X con A.

- Se calcula la ***entropía del nodo N***, o con más precisión del conjunto E_i:

$$\mathbf{H(E_i)= p^+ \log_2(1/p^+) + p^- \log_2(1/p^-)},$$

donde p^+ y p^- representan, respectivamente, las probabilidades (aproximadas por las frecuencias relativas) de clase = + y clase = - en E_i.
(Obsérvese que este cálculo será independiente del atributo X al que se está aplicando el test)

- Se calcula el ***valor medio de la entropía del nivel siguiente***, generado por el atributo X al que se está aplicando el test:

$$\mathbf{H(X/E_i)= p(X= v_1/ E_i)\, H(E_{i1}) + p(X= v_2/ E_i)\, H(E_{i2}) + ...}$$
$$\mathbf{+ p(X= v_j/ E_i)\, H(E_{ij}) + ... + p(X= v_n/ E_i)\, H(E_{in})},$$

donde $v_1, v_2, ..., v_n$ son los valores del atributo X, $p(X= v_n/ E_i)$ representa al probabilidad condicionada de que $X= v_n$ en E_i y E_{ij} es el subconjunto de E_i, $E_{ij} \subset E_i$, tal que sus elementos tienen el valor $X= v_j$.

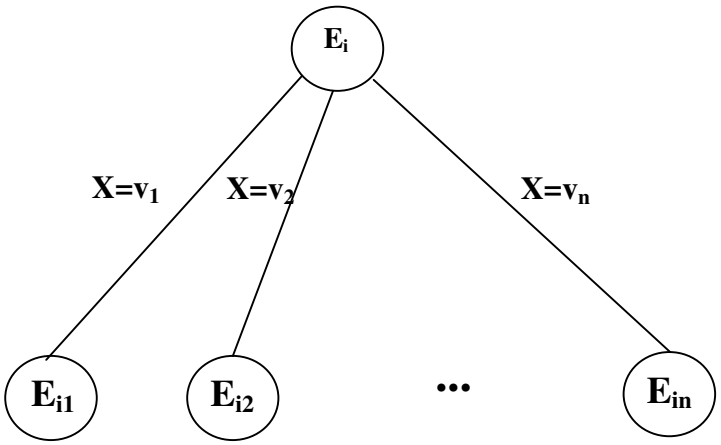
- Se denomina ***ganancia(X)*** a la diferencia de la entropía del nodo N menos la entropía media del siguiente nivel:

$$\mathbf{ganancia(X) = H(E_i) - H(X/E_i)}.$$

- Se elige el ***atributo que maximice la ganancia***.

Con el análisis del **test** queda concluida la exposición de la construcción del árbol de decisión que adquirirá como nuevo conocimiento el sistema.

$$\mathbf{H(E_i)= p^+ \log_2(1/p^+) + p^- \log_2(1/p^-)}$$



Test para el atributo X

$$\mathbf{H(X/E_i)= p(X= v_1/ E_i)\, H(E_{i1}) + p(X= v_2/ E_i)\, H(E_{i2}) + ... + p(X= v_n/ E_i)\, H(E_{in})}$$

Ganancia(X) = H(E_i) - H(X/E_i) **(ID3)**

Razón de Ganacia(X) = Ganancia(X) / Información de Ruptura(X/E_i) **(C4.5)**

$$\mathbf{Información\ de\ Ruptura(X/E_i) = \sum_{j=1}^{n} p(X= v_j/ E_i) \log_2(1/ p(X= v_j/ E_i))}$$

Figura 28.

- **Un ejemplo de aprendizaje**

A continuación desarrollaremos esquemáticamente algunas de las partes más características del proceso de construcción del árbol para un conjunto de ejemplos sencillo.

Ejemplo 3.5. Considérese el conjunto, E, de ocho ejemplos que figuran en la siguiente tabla, descritos por los valores de cuatro atributos y una de las dos posibles clases:

Ejemplos	*Estatura*	*Pelo*	*Ojos*	*Clase*
$\mathbf{e_1}$	Baja	Rubio	Azules	+
$\mathbf{e_2}$	Alta	Rubio	Castaños	-
$\mathbf{e_3}$	Alta	Pelirrojo	Azules	+
$\mathbf{e_4}$	Baja	Moreno	Azules	-
$\mathbf{e_5}$	Alta	Moreno	Azules	-
$\mathbf{e_6}$	Alta	Rubio	Azules	+
$\mathbf{e_7}$	Alta	Moreno	Castaños	-
$\mathbf{e_8}$	Baja	Rubio	Castaños	-

Se inicia el proceso etiquetando la raíz con el conjunto total de ejemplos, E. Inmediatamente se observa que contiene ejemplos de las dos clases existentes, luego es necesaria su expansión. Se realizará el **test(X)** con X recorriendo cada uno de los atributos disponibles, que en este fase son todos los que intervienen en la definición de los ejemplos. Detallaremos el cálculo de la ***ganancia*** para el atributo **color de pelo** en la raíz.

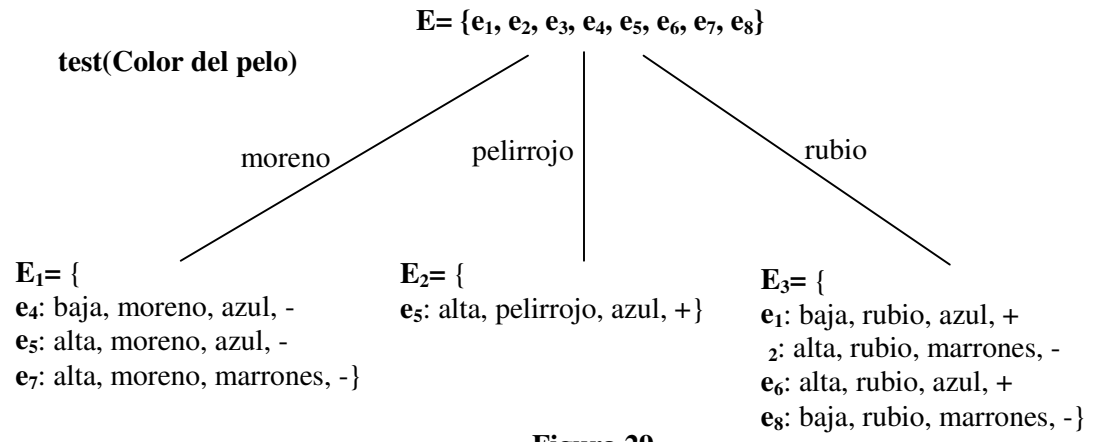


Figura 29.

Puesto que en el conjunto E hay tres ejemplos con la clase + y cinco con la clase -, la entropía en la raíz, que es independiente del atributo que se estudia, será:

$$H(E)= 3/8 \log_2(8/3) + 5/8 \log_2(8/5) = 0.954 \text{ bits}$$

El conjunto E queda particionado de acuerdo con los valores del atributo color_de_pelo, **figura 29**, en los conjuntos:

E_1 ={ejemplos de E que tienen color_de_pelo = moreno},
E_2 {ejemplos de E que tienen color_de_pelo = pelirrojo}
E_3 ={ejemplos de E que tienen color_de_pelo = rubio}.

Seguidamente se calcula la entropía en el nivel de los hijos de la raíz, originados por la expansión provisional correspondiente al atributo color_de _pelo.

H(Color_de_pelo / E) =
p(Color_pelo = moreno / E)H(E_1) +
p(Color_de pelo = pelirrojo / E) H(E_2) +
p(Color_de_pelo = rubio / E) H(E_3).

Donde:

$H(E_1)= p^+\log_2 (1/p^+) + p^-\log_2 (1/p^-) = 0 \log_2(1/0) +1 \log_2(1/1) = 0 + 0 = 0$
$H(E_2)= p^+\log_2 (1/p^+) + p^-\log_2 (1/p^-) = 1 \log_2(1/1) +0 \log_2(1/0) = 0 + 0 = 0$
$H(E_3)= p^+\log_2 (1/p^+) + p^-\log_2 (1/p^-) = 2/4 \log_2(4/2) +2/4 \log_2(4/2) = 0.5 + 0.5 = 1.$

Luego:

H(Color_de_pelo / E) = 3/8 0 + 1/8 0 + 4/8 1 = 0.5 bits,

y por tanto:

ganancia(color_de_pelo) = 0.954 - 0.5 = **0.454 bits**

De manera semejante se calculan:

ganancia(estatura) = **0.003 bits**
ganancia(color_de_ojos) = 0.347 bits.

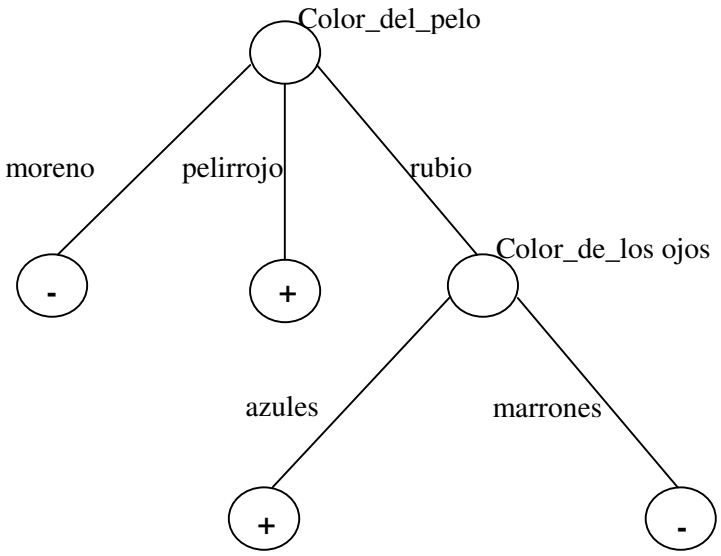


Figura 30.

Lo que determina al atributo color_de_pelo como el que maximiza la ganancia y por tanto con el que se realiza la expansión definitiva del nodo raíz. Los nodos E_1 y E_2 tienen cada uno todos los ejemplos que los etiquetan en una única clase por lo que son hojas y no se expanden. E_3 se expandirá mediante uno de los atributos estatura o color_de_ojos, que se elegirá por el mismo procedimiento. Resultará elegido en esta ocasión el atributo color_de_ojos. Después de la expansión con ese atributo todos los nodos que se tienen serán hojas y se habrá finalizado. El resultado es el árbol de decisión de la **figura 30.**

3.3.1.2. Consideraciones generales acerca del ID3

- **La búsqueda en el espacio de hipótesis en ID3**

El espacio de hipótesis para el este sistema está constituido por todos los árboles de decisión posibles.
Es un espacio *completo* en tanto contiene a todos los árboles con todas las combinaciones discretas posibles de valores de los atributos.

La búsqueda realizada para encontrar la *mejor* hipótesis es *incompleta* en tanto que hay árboles que no se explorarán, sin garantía de que no sean la solución.

- **El *bías* inductivo en ID3**

Es un *bias* de preferencias, frente al *bias* basado en restricciones de otros sistemas. Tiene preferencia por los árboles cortos frente a los más largos. Tiene preferencia por los árboles que sitúan los atributos con mayor ganancia de información cerca de la raíz.

Las preferencias de ID3 siguen el argumento de *la navaja de Occam* (1.320):

Se tiende a preferir las hipótesis sencillas frente a las complicadas

3.3.1.3. El sistema C4.5

A lo largo de este apartado seguiremos básicamente la obra de Quinlan [QUINLAN 93] que documenta la versión comercial de este sistema.

- **El criterio de la ganancia de ruptura**

El test basado en el criterio de maximizar la *ganancia* tiene un sesgo hacia la elección de atributos con muchos valores. Esto es debido a que, obviamente, cuanto más fina sea la partición producida por los valores del atributo, normalmente, la incertidumbre o entropía en cada nuevo nodo será menor, y por tanto también será menor la media de la entropía a ese nivel. C4.5 modifica el criterio de ID3 con ánimo de eliminar este sesgo. Simplemente para un atributo X en la expansión de un nodo N etiquetado con E_i define la ***razón de ganacia*** de X de la siguiente forma:

$$\textbf{Razón de Ganacia(X)} = \textbf{Ganancia(X)} / \textbf{Información de Ruptura(X/E}_i\textbf{)},$$

donde:

$$\textbf{Información de Ruptura(X/E}_i\textbf{)} = \sum_{j=1}^{n} p(X= v_j/ E_i) \log_2(1/ p(X= v_j/ E_i))$$

Quinlan argumenta que efectivamente con este criterio se elimina el sesgo. Como veremos más adelante, hay autores que propone otras fórmulas para obtener mejores clasificaciones con el árbol aprendido.

- **Tratamiento de los atributos continuos**

Tanto el criterio del ID3 como el de C4.5 a la hora de determinar el *mejor* atributo están originalmente pensados para el tratamiento de atributos discretos: bien numéricos (enteros) o bien simbólicos. Sin embargo el comportamiento ideal de un sistema de aprendizaje tiene que ser indiferente ante la característica de discreción o continuidad de los atributos.

El tratamiento que realiza C4.5 de los atributos continuos está basado en el test que estudiamos para los discretos. Si en un nodo N un atributo continuo A presenta valores ordenados v_1, v_2, ... ,

v_n, sencillamente comprueba cuál de los valores $z_i = (v_i + v_{i+1})/2$, $1 \leq i < n$, supone una ruptura del intervalo $[v_1, v_n]$ en dos subintervalos $[v_1, z_i]$ y $[z_i, v_n]$ con mayor ganancia de información, **figura 31.** El atributo A, ahora con únicamente dos valores posibles, entrará en competencia con el resto de los atributos disponibles para expandir el nodo N. Cabe mencionar que ese atributo no deja de estar disponible en niveles inferiores como en el caso de los discretos, aunque con sus valores restringidos al intervalo que domina el camino.

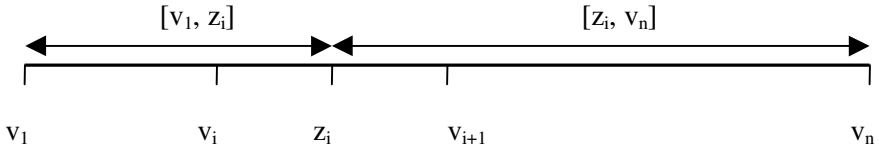


Figura 31.

- **Tratamiento de los valores desconocidos**

C4.5 admite valores de atributos desconocidos en los ejemplos. El tratamiento de estos valores, sin ser difícil de comprender, resulta de larga explicación.
El criterio de ganancia se redefine para el caso de que haya valores de atributos desconocidos, así, para el caso de tratar con un atributo X con valores desconocidos la ganancia al expandir un nodo con etiqueta E_i quedaría:

$$\textbf{ganancia(X) = prob(valor de X sea conocido en } E_i\textbf{) (H(}E_i\textbf{) - H(X/}E_i\textbf{))},$$

donde en el cálculo de $H(X/E_i)$ se tienen en cuenta únicamente los ejemplos con valores de X conocidos. Similarmente se modifica la razón de ganancia introduciendo la modificación previa de la ganancia de ruptura cuyo sumatorio se extiende al conjunto de valores conocidos del atributo X más un sumando que trata el valor desconocido de X (se indicará con ***?***)como un valor más en E_i:

Información de Ruptura(X/E_i) =

$$\sum_{j=1}^{n} \textbf{p(X= }v_j\textbf{/ }E_i\textbf{) log}_2\textbf{(1/ p(X= }v_j\textbf{/ }E_i\textbf{)) + p(X = ?/}E_i\textbf{) log}_2\textbf{(1/ p(X= ?/ }E_i\textbf{)}$$

Una vez realizada la expansión del nodo y obtenidos los conjuntos E_{ij}, que etiquetarán a los nodos hijos en la expansión, el grado de pertenencia de un ejemplo, **e** $\in E_i$, a un conjunto E_{ij}, , ω_{ij}, vendrá definido como:

$$\omega_{ij} = \omega_i \, p(X= v_j),$$

donde ω_i es el grado de pertenencia de **e** a E_i y $p(X= v_j)$ se estima mediante el cociente de la suma de los grados de pertenencia de los ejemplos de E_i con valor de $X= v_j$ dividido por la suma de los grados de pertenencia de los ejemplos de E_i con valor de X conocido.

En la clasificación de un ejemplo mediante el árbol obtenido de la forma anterior, llegados a un nodo con un atributo para el que se desconoce el valor en el ejemplo se siguen todos los caminos posibles, seguramente con diferente grado de pertenencia. Alcanzadas varias hojas la clase del ejemplo se decide comparando la pertenencia del ejemplo a las diferentes hojas. La hoja en la que se alcance el mayor grado decidirá la clase del ejemplo.

- **Control de la expansión del árbol**

Entre las formas de controlar la expansión del árbol existe un control que comprueba la existencia de al menos dos hijos de un nodo cubriendo como mínimo un número de ejemplos

preestablecido; en caso de no verificar esta exigencia el nodo no se expande. Por defecto el sistema toma como número preestablecido el 2. Pueden ser interesantes mayores valores si existe una cantidad apreciable de ruido en el conjunto de ejemplos de entrenamiento.

- **La poda del árbol**

C4.5 efectúa la poda después de haber desarrollado el árbol completo, a diferencia de otros sistemas que realizan la construcción del árbol y la poda a la vez; es decir, estiman la necesidad de seguir desarrollando un nodo aunque no posea el carácter de hoja. Por cualquiera de los métodos se obtienen árboles en los que las hojas pueden cubrir ejemplos de más de una clase y que, por tanto, cometerán errores de clasificación sobre el conjunto de ejemplos de entrenamiento. Precisamente, la estimación del error de clasificación de una hoja, no ya sobre el conjunto de entrenamiento sino sobre ejemplos posteriores a la fase de aprendizaje, decidirá la poda en C4.5. Dicha estimación se efectúa mediante técnicas no muy ortodoxas desde el punto de vista de la Estadística; como el mismo Quinlan comenta [QUINLAN 93]: *...constituyen un gran grano de sal*. Es destacable en el proceso un nivel de confianza que por defecto el sistema toma tan solo del 25%, por ser el nivel que *mejores* resultados proporciona por lo general, aunque permita modificarlo a la hora de ejecutar el programa. El valor de este porcentaje está en relación inversa al grado de la poda que determinará. La poda se decide por comparación del error estimado para un conjunto de hojas, hijas todas de un mismo nodo, y el error estimado para el padre en caso de podar sus hojas. Decidida la poda de las hojas se repite la operación en un nivel superior; así hasta que la poda no proceda.

Sea un cierto nodo hoja que cubre N ejemplos y de ellos E errores. La probabilidad de error de reescritura de la hoja puede aproximarse por el cociente E/N, pero este cociente no es adecuado para estimar la probabilidad de error de la hoja en la clasificación de ejemplos que no pertenezcan al conjunto de entrenamiento. Sin embargo, la probabilidad de error sobre esos otros ejemplos seguirá una distribución de probabilidad que Quinlan resume mediante el intervalo de confianza de la distribución binomial, para el nivel de confianza CF y que por defecto es, como ya se dijo, del 25%. La probabilidad de error de la hoja sobre ejemplos no utilizados en el entrenamiento se identifica con el límite superior de ese intervalo de confianza, $S_{CF}(E,N)$, con el argumento de que el árbol fue construido para minimizar el error. Para simplificar la última fase de este proceso, se supone que se trabaja con un conjunto de N ejemplos nuevos; es decir, el número de ejemplos de entrenamiento que clasificaba la hoja. Así, el número de errores previstos para la hoja será: $N \times S_{CF}(E,N)$. El número de errores previstos para todas las hojas; es decir el subárbol que cuelga de su padre, es la suma de los errores previstos para las hojas. Este último valor se comparará con el número de errores previstos para el padre una vez convertido en hoja al podar provisionalmente todos sus hijos. La comparación de los dos valores decidirá la poda final.

- **La obtención de reglas a partir del árbol**

Es este un proceso largo y complejo. C4.5 extrae reglas del árbol que obtiene previamente a aplicar la poda. Es este un proceder que no es mucho más complejo que los sistemas que extraen las reglas a la vez que elaboran el árbol podado y que produce buenos resultados, argumenta Quinlan.

La lectura de cada camino de la raíz a una hoja en el árbol previo a la poda produce una regla en forma normal conjuntiva, normalmente demasiado específica. Con la familia de reglas resultante de la lectura de todos los caminos se procede en tres fases bien diferenciadas:

- *La cualificación* de las reglas, que consiste en eliminar algunos valores de atributo en el antecedente de cada regla, con lo que se convierten en más generales. Se eliminarán aquellos atributos cuya ausencia del antecedente no signifique mayor merma en la fiabilidad futura de la regla.

La eliminación de atributos se realiza de forma secuencial. Valorado el rendimiento de la regla de partida R y el de la obtenida R^- tras la eliminación de un cierto atributo A en ella, se decide su eliminación definitiva. Si hay pocos atributos se prueba con todos y se elimina el que mejor comportamiento conceda a la regla resultante. Si hay muchos atributos en el antecedente se les ordena por medio de un heurístico y se respeta el orden en el estudio de su posible eliminación; el proceso algorítmico correspondiente es voraz; sin vuelta atrás. Para la valoración de las reglas R y R^- se comparan las probabilidades de error estimadas para cada una; equiparando esas probabilidades con los límites superiores de los intervalos de confianza calculados de forma similar a como se hizo en la poda de árboles.

- *La selección* de una familia de reglas. El conjunto de reglas obtenido en la fase anterior normalmente es redundante como recubrimiento del conjunto de ejemplos. En esta fase se trata de eliminar aquellas reglas *peores* pero dejando como remanente una familia de reglas suficiente para cubrir a la mayoría de los ejemplos.

La selección se efectúa siguiendo el *Principio de Mínima Descripción (MDP)* (Rissanen 1978). En líneas generales consiste en considerar la familia inicial de reglas como una teoría y sus errores como excepciones a la teoría y tratar de buscar una subfamilia que optimice la suma de las codificaciones de la teoría correspondiente y de las excepciones. En las codificaciones de las reglas y de las excepciones hay que realizar algún ajuste debido a la conmutatividad de los atributos del antecedente de cada regla y de los errores en la semántica del sistema *teoría+excepciones*. El procedimiento resulta pesimista por lo que Quinlan mejor que la suma directa de las codificaciones de reglas y excepciones realiza la siguiente combinación lineal:

$$\textit{bits de la excepciones} + \omega\ \textit{bits de la teoría,}$$

donde ω representa un factor menor que uno y que depende de la probabilidad de que dos teorías representen la misma colección de ejemplos. Lo que está íntimamente relacionado con la redundancia de los atributos; es decir, con la relación del número de atributos que se manejan y el número de ellos realmente necesarios para determinar completamente el atributo clase - concepto de *clave* en bases de datos. Por defecto el sistema trabaja con $\omega = 0.5$, sin embargo Quinlan aclara que: ... *por suerte el sistema no es muy sensible a este parámetro.*

- *La regla por defecto*. La regla por defecto, regla que carece de antecedente, se introduce para clasificar a los ejemplos que no quedan cubiertos por la familia de reglas de la fase anterior. El consecuente es la clase más numerosa entre los ejemplos que restan por cubrir.

Todavía después de haber introducido la regla por defecto se realiza una nueva depuración eliminando reglas con bajo rendimiento aunque, para mejorar la comprensión y coherencia de la familia de reglas resultante en muchas ocasiones, se mantiene reglas que son del todo superfluas para la correcta clasificación de los ejemplos de entrenamiento.

Todas las fases están pensadas para obtener una buena fiabilidad de la familia de reglas final sobre conjuntos de ejemplos diferentes al de entrenamiento.

- **La agrupación de valores de atributos discretos**

Cuando el dominio de un atributo discreto es muy grande el criterio de la razón de ganancia puede mostrar un cierto sesgo contrario a este tipo de atributos, ya que la información de ruptura tiene posibilidades de hacerse muy grande (véase la entropía máxima de una fuente de memoria nula) y reducir excesivamente la razón de ganancia. Es éste uno de los motivos por los que puede resultar interesante que el sistema agrupe valores de atributos discretos. La elección de esta opción se realiza de una forma empírica pues se trata de decidir cuando hay presentes atributos discretos con *muchos* valores. Es digno de destacar que en el caso de un conjunto de ejemplos famoso en la literatura, el problema *Monk's 3*, con un 5% de ruido, únicamente

optando por agrupar valores en sus atributos consigue C4.5 alcanzar la tasa del 0% de errores sobre el conjunto test. Algo que es difícil de alcanzar mediante el ajuste de otros parámetros.

El proceso de agrupación, una vez elegida esta opción que por defecto no está presente, se realiza en la expansión de cada nodo en el que se realice el *test* para un atributo con valores discretos. El proceso busca el par de grupos de valores del atributo que optimice la razón de ganancia, para ello realiza una unión iterativa de grupos de valores. Comienza con los valores individuales y en cada iteración se valora la unión de todos los pares posibles de grupos por medio del criterio de la razón de ganancia. El par ganador se une y se comienza una nueva iteración. Así hasta que no haya ningún par de grupos que mejore la razón de ganancia o hasta que queden dos grupos de valores.

- **El funcionamiento por ventanas**

Es una forma de funcionar heredada del ID3, pero que perdió parte del sentido con C4.5. Está pensada para manejar conjuntos de ejemplos de entrenamiento tal que el manejo en su totalidad en la fase de aprendizaje sobrepase la capacidad del sistema. A esta situación era fácil de llegar con los ordenadores de los años setenta, cuando vio la luz ID3. Actualmente la capacidad de los ordenadores puede considerarse suficiente para manipular conjuntos de ejemplos del tamaño que habitualmente se entiende por grandes; además, para conjuntos de ejemplos muy grandes, como los que suelen darse cuando se manipulan técnicas de aprendizaje para *explotar* ingentes cantidades de ejemplos provenientes de grandes bases de datos (*Data Mining*), existen técnicas añadidas que permiten realizar depuraciones de los ejemplos o compactación semántica de la información. Así es que la utilización del funcionamiento por ventanas en C4.5 está justificado, como veremos, por otros motivos diferentes.

Este método de ventanas comienza con la elección aleatoria de un subconjunto de ejemplos, uniformemente distribuidos respecto a las clases, de entre disponibles para el entrenamiento. Dichos ejemplos se utilizan en una primera fase de aprendizaje o *ventana*. Con el árbol o conjunto de reglas resultantes se realiza una clasificación de los ejemplos que han quedado como remanentes en la fase anterior. Los ejemplos que resulten mal clasificados en este proceso se mantienen en el conjunto de remanentes mientras que se prescinde definitivamente de los que hayan resultado bien clasificados. Con los ejemplos utilizados en la ventana anterior más una cantidad limitada de los remanentes se efectúa otro proceso de aprendizaje y se vuelve a refinar el conjunto de ejemplos remanentes de forma semejante a la primera vez. Se realizan sucesivas iteraciones hasta que el proceso termina cuando el conjunto de ejemplos remanentes sea suficientemente pequeño o se haya sobrepasado un número de ventanas preestablecido.

Obsérvese que este método está basado en el principio siguiente: en cada iteración se prescinde definitivamente de los ejemplos no incluidos en la última de las fases de aprendizaje realizadas que resultan bien clasificados por el árbol, o un conjunto de reglas, porque se supone que están bien *representados* por los ejemplos que sí han intervenido en el aprendizaje.

Puesto que la ventana inicial se obtiene aleatoriamente el resultado final puede diferir de una ejecución a otra. Este es uno de los motivos de interés de las ventanas en C4.5, pues permite obtener muchas reglas diferentes como unión de las reglas obtenidas en sucesivas ejecuciones. Otro motivo es porque la experiencia demuestra que hay muchas ocasiones en las que el árbol o reglas resultantes mejoran la precisión de clasificación del árbol o reglas obtenidos en ejecuciones estándar.

El funcionamiento por ventanas se regula por medio de diferentes parámetros que permiten regular el número de árboles máximo, el porcentaje de los ejemplos remanentes que se incorporan a la siguiente ventana, etc.

- **Métodos de validación de los resultados del aprendizaje**

En un principio, para la validación de los sistemas se aceptaba como suficiente el comportamiento del conocimiento adquirido sobre el conjunto de entrenamiento, proporcionándose como una medida de la fiabilidad del sistema el porcentaje de los errores de clasificación sobre este conjunto, denominado porcentaje de errores de reescritura. No se necesita de mayores comentarios, después de lo desarrollado hasta aquí, para comprender que este método de validación tiene muy poco rigor. La utilización de un de test incluyendo ejemplos que no intervinieron en el aprendizaje para validar un sistema resultó un procedimiento más convincente, siempre que se dispusiese de los ejemplos necesarios para este menester. Si inicialmente se tiene un conjunto de ejemplos disponibles suficientemente numeroso siempre es posible dividirlo en dos subconjuntos; uno para el entrenamiento y otro para el test. Esta idea, generalizada incluso para el caso en que no se tuvieran muchos ejemplos, constituye el método conocido *cross validation*, consistente en dividir, uniformemente respecto de las clases, los ejemplos disponibles en N subconjuntos de cardinal lo más parecido posible. A continuación se realizan N procesos de aprendizaje excluyendo en cada vez uno de los N subconjuntos de ejemplos anteriores del conjunto de entrenamiento, que estará formado por los ejemplos de los otros N-1 subconjuntos. En cada ocasión se calcula el porcentaje de errores del conocimiento adquirido sobre el conjunto de ejemplos excluido en la fase de aprendizaje. Finalmente, se calcula la media de los errores en cada uno de los N procesos, valor que se considera como medida de la precisión del sistema. Para el caso de disponer de pocos ejemplos la elección de N que proporciona mayor fiabilidad es la de un N igual al número de ejemplos; en cada proceso de aprendizaje se aprende de todos los ejemplos menos uno y se prueba el rendimiento sobre el excluido.

El software de C4.5 dispone de una utilidad para realizar la validación por medio del *cross validation*.

- **Limitaciones**

La limitación más importante de C4.5 es la aproximación a base de *rectángulos* que efectúa de las clases. En la **figura 32** se observa como los valores de dos atributos determinan rectángulos, que son las *unidades espaciales* que permiten *definir* las clases. Los atributos A y B con sus valores se representan en dos ejes en los que la ordenación no tiene ningún sentido especial. Admitamos que las posiciones de los ejemplos denotan una cierta relación de situación entre ellos. Se observa como A= v_1 y B= v_1 determinan un rectángulo que contiene solamente elementos de la clase -; igualmente el rectángulo delimitado por A= v_1 y B= v_2 contiene ejemplos que son todos de la clase + menos uno, que es de la clase -, y por tanto la clasificación con estos dos valores produciría un error de reescritura.

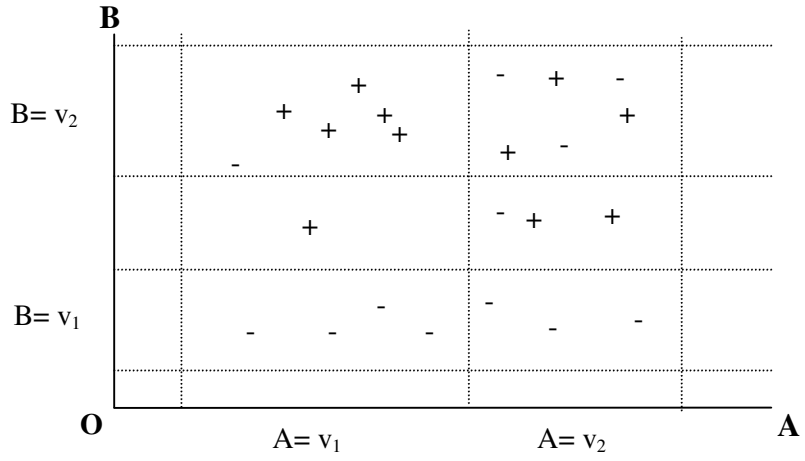


Figura 32.

El rectángulo A= v_2 y B= v_2 contiene igual número de ejemplos de las dos clases que salvo algún refinamiento de esos valores de atributo -si fuera posible refinarlos- no se podrán obtener rectángulos que clasificasen bien a todos los ejemplos que caen en su interior. Esa es la gran limitación de los árboles de decisión y en especial de C4.5. Para que realmente alcance una solución relativamente sencilla las clases tienen que ser *separables* mediante rectángulos.

Una consecuencia práctica de la comprensión de esta limitación es que cuando no se consigue rebajar el número de errores una posibilidad es desarrollar exhaustivamente el árbol con todos los atributos disponibles para que los rectángulos sean más lo más *pequeños posible* y así conseguir mayor tasa de aciertos. Recordemos que el principal control del desarrollo del árbol se tiene prefijando el número mínimo de ejemplos abarcados por al menos dos hijos del nodo que se expande.

3.3.2. MÉTODOS ALTERNATIVOS PARA EL APRENDIZAJE DE ÁRBOLES

3.3.2.1. EL PROBLEMA DE LA FRAGMENTACIÓN

Un problema característico de los sistemas que, a semejanza de C4.5 proporcionan árboles cuya construcción está basada en sucesivas subdivisiones del conjunto de ejemplos mediante un test relativo a un solo atributo, es que repitan subárboles en su estructura; es decir, estos árboles no son la forma más económica de representar el sistema de clasificación que encierran. Este fenómeno es conocido como el ***problema de la fragmentación***. Suele hacerse patente cuando se tratan de adquirir conceptos que admiten una definición en forma normal disyuntiva de sus atributos con pocas cláusulas. La extracción de reglas a partir del árbol es una solución, pero hay otras soluciones que tratan de evitar la duplicación inicial por medio de otras estructuras más ricas.

- **Grafos de decisión.**

Una de las primeras soluciones intuitivas que se nos ocurren al *problema de la fragmentación* es la construcción de grafos que no sean árboles como representación del conocimiento; pues bien, hay soluciones por esa vía (Oliver et al., 1992 y Kohavi 1994).

- **Árboles con múltiples expansiones en cada nodo.**

Otra solución al problema anterior es la aplicación de test extendidos a más de un atributo; incluso algún autor propone realizar test que se correspondan con funciones linealmente separables.

3.3.2.2. CRITERIOS ALTERNATIVOS PARA ELEGIR *EL MEJOR* ATRIBUTO

Después de ID3 hubo muchas propuestas para mejorar el test de Quinlan o simplemente alternativas. A continuación recogemos algunas de las variantes que mayor impacto tuvieron.

- Criterio basado en la minimización del ***índice de diversidad de Gini*** de un atributo X:

$$\sum_{i.j \in dom(X)} p(X = v_i)\, p(X = v_j) = \sum_{i \in dom(X)} p(X = v_i)^2$$

- ***Medida de Schlimmer*** (1990) ***y Tan*** (1993) proponen tomar como criterio de selección del mejor atributo X la razón:

$$\frac{\text{Ganancia}^2(X)}{\text{Coste}(X)}$$

donde Coste(X) es un coste asignado al atributo X. Los autores lo aplicaron con éxito en la percepción de un robot.

- Criterio de la ***ganancia normalizada***, propuesta por Hwan, Soo, Song y Kim. Se trata de aplicar el criterio de ganancia con una base logarítmica *n*, donde *n* es igual al número de valores que presenta el atributo que se está evaluando, y dividir la ganancia así obtenida por $\log_2 n$:

$$\frac{\text{Ganancia}(X,\log_n)}{\log_2(n)}$$

  Según sus autores en ciertos problemas mejora el criterio de ganancia y el de la razón de ganancia.

- ***Medida de Núñez*** [NÚÑEZ, 88]; facilita el manejo de costes proporcionándoles mayor o menor importancia según las necesidades por medio del ajuste de $\omega \in [0,1]$:

$$\frac{2^{\text{Ganancia}(X)} - 1}{(\text{Coste}(X)+1)^{\omega}}$$

- Criterio basado en al ***distancia entre particiones de López de Mántaras*** [LÓPEZ DE MÁNTARAS 91]. Esta medida permite evaluar las distancias de la partición generada por un atributo a la partición generada por el atributo clase. Se elegirá el atributo que minimice esa distancia.

- Criterio basado en la ***aplicación del Test Chi Cuadrado***.

3.3.2.3. OTRAS SOFISTICACIONES

- **Utilización de conocimiento previo**

Nuñez (Machine Learning 6 (3):231-250, 1991) propone un método para desarrollar un árbol más compacto partiendo de jerarquías *is_a* así como conjunciones de atributos entre nodos de decisión.

- **Descubrimiento e incorporación de ejemplos compuestos**

El objetivo de este procedimiento es disminuir la complejidad del árbol a base de formar términos compuestos como conjunción de atributos. El método ocasiona el problema de que la misma conjunción de atributos se repita por el árbol. En esta línea trabaja el sistema FRINGE del que volveremos a hablar.

- **Introducción del operador relacional $\neq$**

Fiol [FIOL, 91]) introduce el concepto de árbol mínimo. En su construcción utiliza el operador relacional $\neq$ con lo que gana mayor expresividad en el lenguaje del conocimiento extraído, tanto

árboles como reglas. En términos más formales consiste en admitir la negación de los literales que intervienen en las ramas o en los antecedentes de las reglas.

- **Incrementalidad**

El más claro representante de la incrementalidad en el aprendizaje de árboles de decisión es el sistema ID5R de Utgoff, [UTGOFF, 89] que constituye una versión incremental y eficiente del ID3. Una descripción general del proceder de este sistema puede verse en [LANGLEY 96].

- **Otros tratamientos de los atributos continuos**

Se han estudiado infinidad de tratamientos de atributos continuos acudiendo y sin acudir al concepto de entropía.

3.3.3.. BREVE DESCRIPCIÓN DE OTROS SISTEMAS DE APRENDIZAJE DE ÁRBOLES

Ya por último nos resta realizar una visión muy panorámica de otros sistemas que aprenden árboles de decisión. Hemos seleccionado tres de los más representativos en la literatura, pero hay muchos más:

- **CART (Breiman et al. 1984)**

Este sistema construye árboles de regresión que aprende una función de valores reales. Las hojas del árbol está etiquetadas con valores numéricos reales frente a las clases en los árboles de decisión. Su estrategia de funcionamiento es dividir el conjunto de entrenamiento en polígonos que no se solapen. Las sucesivas divisiones producen un árbol binario Los datos pueden ser cualitativos o cuantitativos. En cada paso el objetivo es independiente del anterior y se busca dividir el conjunto de ejemplos en la mejor forma de encajar con las clases. Una vez finalizada esta fase realiza la poda, siendo este tratamiento original de este sistema. En cada nodo CART es capaz de obtener una forma disyuntiva ($s_1 \wedge s_2 \ ... \wedge s_2$) o conjuntiva ($s_1 \vee s_2 \ ... \vee s_2$), donde s_i es una *división* de un atributo primitivo.

- **ASSISTANT (Kononenko y Bratko 1984)**

No construye el árbol iterativamente a semejanza de ID3, sino que elige un subconjunto de ejemplos representativos del conjunto de entrenamiento; otra diferencia es que no va subdividiendo el conjunto de ejemplos en subconjuntos disjuntos, de forma que permite que una clase sea una parte de otra más amplia. La poda del árbol según se construye va orientada hacia la exactitud de la clasificación sobre el conjunto test. Por último, cabe comentar que existe una versión comercial de este sistema y, que nosotros sepamos, no está disponible para uso académico.

- **FRINGE (Pagallo et al. 1990)**

Construyen un árbol al estilo *clásico* para posteriormente definir nuevos conceptos intermedios que sirvan para simplificar la expresión de árbol inicial. Actúa iterativamente hasta que no logra definir nuevos conceptos. Es una solución al *problema de la fragmentación*. Fue posteriormente extendido para manejar atributos continuos (Pagallo 1989)[2].

[2] La aparente contradicción de fechas quizás se deba a la demora en la publicación del artículo de 1990 en *Machine Learning*.

3.4. APRENDIZAJE DE REGLAS CON VARIABLES: *FOIL*.

Uno de los sistemas pioneros en el aprendizaje inductivo de reglas con variables -lógica de 1º orden- es FOIL de Quinlan. Numerosos sistemas posteriores son proclamados por sus autores como descendientes de FOIL; este es el caso de FOCL que también se estudiará más adelante, dentro de los sistemas clasificados como sistemas integrados o de revisión de teorías.

- **Generalidades**

FOIL desciende y comparte autor con ID3 y utiliza, a semejanza de este último, una medida basada en la información para decidir el atributo que se debe de elegir en cada momento para completar una regla. Su proceder es, pues, de lo general a lo particular. Sus entradas consisten en un conjunto de predicados o relaciones expresadas extensionalmente; es decir, por las tuplas que las integran. A su vez, las tuplas de cada relación se clasifican en positivas o negativas según ejemplifiquen o no al predicado objetivo, cuya definición en forma de regla y a partir de los otros predicados se trata de captar. Asume la hipótesis del mundo cerrado, por la que toda tupla que no aparezca definida explícitamente como positiva se interpreta como negativa. El conocimiento aprendido lo expresa en la lógica de las cláusulas Horn con la notación de Kowalski, sin símbolos de función pero con literales negativos; aprende, pues, relaciones tales como:

nieta(x,y) ← padre(y,z), padre(x,x), mujer(y)

En la regla anterior el predicado o relación objetivo es *nieta* y viene definido por las relaciones *padre* y *mujer*.

En su estrategia de funcionamiento parte de la regla más general con el predicado objetivo como consecuente y sin antecedente, -p.e. *nieta(x,y)* ←, en el caso de la regla anterior- y le va añadiendo los predicados a su antecedente que considera óptimos en el momento de su elección, según una técnica de decisión heurística. De esta forma prosigue hasta obtener una regla que cubra solamente ejemplos positivos. Al igual que los algoritmos de recubrimiento, tras eliminar los ejemplos cubiertos por la regla recién construida continua el cálculo de la siguiente a partir de los ejemplos positivos remanentes. Así hasta alcanzar una familia de reglas que cubra solo a los ejemplos positivos. Los ejemplos negativos se clasifican por defecto.

- **Algoritmo**

FOIL (Predicado Objetivo, Predicados, Ejemplos)

Comenzar

- Comprueba la existencia de relaciones binarias simétricas para evitar una ordenación redundante de las variables.
- Ordena las constantes para prevenir la no terminación desde cualquier cláusula recursiva.
- Inicializa Positivos con las tuplas para las que el predicado objetivo es Verdad*/
- Inicializa Negativos con las tuplas para las que el predicado objetivo es Falso*/
- Reglas_aprendidas:= [ ]
- **mientras** Positivos ≠ { } hacer
  - */*Aprende_una_NuevaRegla*/*
  - NuevaRegla:= Regla que predice el predicado objetivo sin antecedente
  - NuevaReglaNegativa:= Negativos
  - **mientras** NuevaReglaNegativa ≠ { } hacer
    - */*Añadir un nuevo literal para especializar NuevaRegla*/*
    - Literales_candidatos:= nuevos literales candidatos para NuevaRegla, generados basándose en Predicados
    - Mejor_literal:= L∈ Literales_candidatos: GananciaFoil(L, NuevaRegla) es máxima
    - Añadir Mejor_literal al antecedente de NuevaRegla
    - NuevaReglaNegativa:= subconjunto de NuevaReglaNegativa que no satisfacen el antecedente de NuevaRegla