Aprendizaje de árboles de decisión

Curso: Minería de Datos

Universidad Nacional de San Luis (UNSL)
San Luis - Argentina
Año 2018

1. Introducción

El aprendizaje de árboles de decisión es un método muy simple, que ha sido ampliamente utilizado y con gran éxito en numerosas tareas de aprendizaje inductivo. Es un método de aproximación de funciones robusto a la presencia de datos erróneos y es capaz de aprender expresiones disyuntivas. Existe toda una familia de algoritmos de aprendizaje de árboles de decisión que incluye a algoritmos muy conocidos como ID3, ASSISTANT y C4.5. Esta familia de algoritmos, referenciada a veces como TDIDT (*T*op-*D*own *I*nduction of *D*ecision *T*rees) se caracteriza por buscar en un espacio de hipótesis completamente expresivo que evita las dificultades de los espacios de hipótesis restringidos. Su sesgo inductivo es un sesgo de preferencia por árboles pequeños sobre árboles grandes.

2. La representación de árboles de decisión

En los algoritmos TDIDT, la función aprendida es representada mediante un *árbol de decisión*. Estos árboles pueden ser convertidos en conjuntos de reglas $if - then$ si se desea obtener una mejor legibilidad. En la figura 1 se muestra un árbol de decisión típico, para el concepto de $JugarTenis$. Este árbol de decisión clasifica días de la semana de acuerdo a si son adecuados para jugar tenis o no.

Cada nodo interno en el árbol corresponde a un test del valor de algún *atributo* (o propiedad) de la instancia a clasificar, y las ramas que descienden de este nodo son rotuladas con los valores posibles del test. Cada nodo hoja de un árbol de decisión especifica el valor retornado en caso de que dicha hoja sea alcanzada. Estos valores corresponden a posibles clasificaciones de una instancia.

Una instancia se clasifica comenzando en el nodo raíz del árbol, testeando el atributo especificado por este nodo, y moviéndose hacia abajo por la rama del árbol que corresponde al valor del atributo en la instancia a clasificar. Este proceso se repite para el subárbol cuya raíz es el nuevo nodo, y así sucesivamente hasta alcanzar un nodo hoja, en cuyo caso se retorna la clasificación asociada con este nodo. Por ejemplo, siguiendo este proceso la instancia:

$$\langle Estado = soleado, Temperatura = caluroso, Humedad = alta, Viento = verdadero \rangle$$

será clasificada como una instancia negativa por el árbol de decisión (es decir, el árbol predice que $JugarTenis = no$).

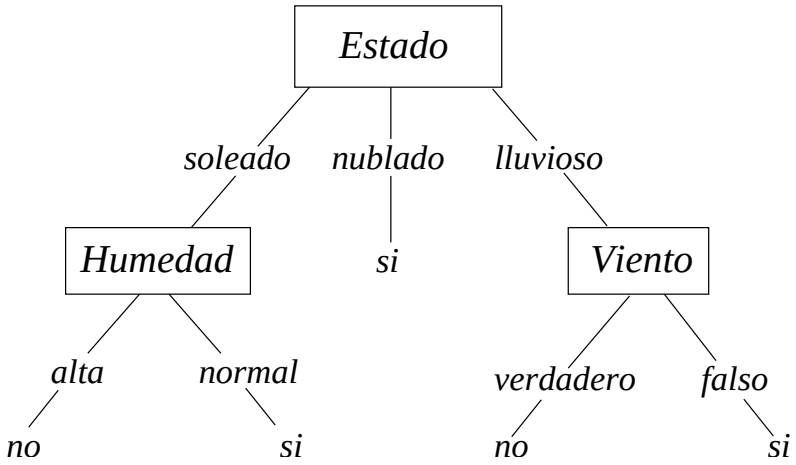


Figura 1: Un árbol de decisión para el concepto $JugarTenis$.

En general, los árboles de decisión representan una disjunción de conjunciones de restricciones sobre los valores de atributo de las instancias. Cada paso desde la raíz del árbol a una hoja, corresponde a una conjunción de tests de atributos, y el árbol en su conjunto a una disjunción de estas conjunciones. Por ejemplo, el árbol de la figura anterior corresponde a la expresión:

$$(Estado = soleado \ \wedge \ Humedad = normal)$$
$$\vee$$
$$(Estado = Nublado)$$
$$\vee$$
$$(Estado = lluvioso \ \wedge \ Viento = falso)$$

3. Problemas apropiados para el aprendizaje de árboles de decisión

Aún cuando existen diversos métodos para aprender árboles de decisión, este tipo de aprendizaje se adapta en general a problemas con las siguientes características:

- Las instancias son representadas por pares atributo-valor.
- La función objetivo tiene valores de salida discretos.
- Pueden requerirse descripciones disyuntivas.
- Los datos de entrenamiento pueden contener errores (en la clasificación de los ejemplos de entrenamiento o en los valores de sus atributos).
- Los datos de entrenamiento pueden tener valores de atributos desconocidos.

Los problemas prácticos que cumplen con estas características son innumerables, como lo han sido las aplicaciones de árboles de decisión para su resolución. En la mayoría de estos casos, la tarea consistió en clasificar los ejemplos dentro de un conjunto discreto de categorías posibles, lo que se denomina comúnmente como *problema de clasificación*.

4. Algoritmo básico para el aprendizaje de árboles de decisión

El problema de encontrar un árbol de decisión "consistente" con los ejemplos de entrenamiento puede parecer dificultoso, pero en realidad hay una solución trivial. Uno podría construir un árbol que tiene un paso a una hoja *por cada ejemplo*, donde el paso testea cada atributo y sigue el valor correspondiente al ejemplo y la hoja tiene la clasificación del ejemplo. Si el mismo ejemplo es presentado nuevamente, el árbol de decisión dará la clasificación correcta. Lamentablemente, este árbol no tendrá mucho para decir sobre ningún ejemplo por fuera del conjunto de entrenamiento, ya que sólo se ha dedicado a *memorizar* las observaciones. Dado que no se ha realizado ningún intento por extraer ningún patrón desde los ejemplos, no podemos esperar que este árbol trivial sea capaz de extrapolar a ejemplos que nunca ha visto.

Otra idea, consistiría en aplicar la *Ockham's razor*[1] y encontrar el árbol de decisión *más pequeño* que sea consistente con los ejemplos. Dado que esta tarea puede resultar intratable, una alternativa es utilizar una heurística simple que haga una buena tarea en encontrar árboles "bastante" pequeños que sean consistentes con los datos de entrenamiento.

Para llevar esta última idea a la práctica, un algoritmo de aprendizaje de árboles de decisión debería intentar testear el atributo *más importante* primero. En este caso, por "más importante" significamos que el atributo hace la mayor contribución para la clasificación de un ejemplo. De esta manera, si repetimos el mismo razonamiento con todos los atributos, esperaremos conseguir la clasificación correcta con un número pequeño de tests, significando que todos los pasos en el árbol serán cortos y el árbol en su conjunto será pequeño.

La mayoría de los algoritmos para aprender árboles de decisión, se basan en un algoritmo central que realiza una búsqueda *greedy*, "top-down" a través del espacio de posibles árboles de decisión. Estos algoritmos, toman en cuenta un conjunto de ejemplos de entrenamiento E, un atributo objetivo $A_{\mathcal{O}}$ cuyo valor debe predecir el árbol, un conjunto de atributos $Atributos$ que pueden ser testeados por el árbol de decisión y dan como salida un árbol de decisión que clasifica correctamente las instancias en E. Para el caso del algoritmo ID3, una versión simplificada en pseudo-código se muestra en la figura 2.

Como podemos observar, el caso más trivial para tratar por el algoritmo de aprendizaje es aquel en que todos los ejemplos de entrenamiento pertenecen a la misma clase (tienen el mismo valor para el atributo objetivo) que se muestra en el paso 2 de la figura 2. En este caso, el algoritmo se limita a construir un nodo hoja cuyo valor v_j es el valor en $V(A_{\mathcal{O}})$ común a todos los ejemplos de entrenamiento.

Si los ejemplos de entrenamiento tienen distintas clasificaciones, significa que para clasificar las instancias en E tendremos que recurrir a algún test de los atributos. En este caso, puede darse la situación de que no existan atributos para testear, por lo cual, lo mejor que se puede hacer es retornar como clasificación aquel valor en $V(A_{\mathcal{O}})$ que sea más común en E (paso 3).

Si $Atributos \neq \emptyset$, el algoritmo de aprendizaje debe decidir cual es el atributo $A \in Atributos$ que utilizará para clasificar las instancias. Este aspecto es crucial en este tipo de aprendizaje y será analizado en más detalle en la sección 4.1. Por el momento, sólo es necesario observar que si $|V(A)| = n$ es el número de valores posibles de A, el conjunto E puede ser particionado en n subconjuntos mutuamente

[1] Principio filosófico que sugiere seleccionar la hipótesis más simple que se ajusta a los datos.

función $ID3(E, A_{\mathcal{O}}, Atributos)$
entrada: E, un conjunto de ejemplos de entrenamiento.
$A_{\mathcal{O}}$, el atributo cuyo valor el árbol debe predecir.
$Atributos$, el resto de los atributos que pueden ser testeados por el árbol de decisión.
salida: Un árbol de decisión.

1) Crear un nodo *Raíz* para el árbol
2) Si $\forall e \in E$, $e(A_{\mathcal{O}}) = v_j$, retornar el árbol de un único nodo *Raíz*, con rótulo v_j
3) Si $Atributos = \emptyset$, retornar el árbol de un único nodo *Raíz*, con rótulo $mcm(A_{\mathcal{O}})$[a]
4) $A \leftarrow$ el atributo que *mejor*[b] clasifica E
5) *Raíz* $\leftarrow A$
6) Por cada valor posible, $v_i \in V(A)$,
6.a) Agregar una nueva rama debajo de *Raíz*, correspondiente al test $A = v_i$
6.b) Sea E_{v_i} el subconjunto de E que tienen valor v_i para A
6.c) Si $E_{v_i} = \emptyset$
6.c.1) entonces debajo de esta nueva rama agregar un nodo hoja con rótulo $mcm(A_{\mathcal{O}})$
6.c.2) sino debajo de esta nueva rama agregar el subárbol dado por
$ID3(E_{v_i}, A_{\mathcal{O}}, Atributos - \{A\})$
7) retornar *Raíz*
fin_función

[a] $mcm(A_{\mathcal{O}})$ es el valor más común de $V(A_{\mathcal{O}})$ en E.
[b] El mejor atributo es aquel con más alta ganancia de información.

Figura 2: Algoritmo de aprendizaje ID3

disjuntos E_{v_1}, $E_{v_2}, \ldots, E_{v_n}$ tal que E_{v_i} es el subconjunto de E que tienen valor v_i para A. En este caso, el algoritmo de aprendizaje retornará un árbol de raíz A que tendrá una rama rotulada v_i por cada $v_i \in V(A)$. Debajo de cada rama con rótulo v_i, tendremos el subárbol de decisión que resulta de aplicar el algoritmo de aprendizaje en forma recursiva sobre el subconjunto de ejemplos E_{v_i} y el resto de los atributos disponibles (pasos 5, 6 y 7).

4.1. La elección del atributo

El aspecto central en el algoritmo ID3, es la elección del atributo a testear en cada nodo del árbol. Uno querría seleccionar el atributo que es *más útil* para clasificar los ejemplos. Supongamos por el momento que nuestra intensión es lograr *árboles de decisión* tan pequeños como sea posible, y que clasifiquen adecuadamente los ejemplos. Si observamos el algoritmo de la figura 2, podemos observar que es un algoritmo recursivo cuyo punto de parada principal se produce cuando *todos* los ejemplos de entrenamiento tienen el mismo valor para el atributo objetivo. En este caso, podríamos decir que el conjunto de ejemplos de entrenamiento es totalmente "puro"(u homogéneo). Dado que nosotros deseamos obtener árboles pequeños, la idea en este caso, sería detener el proceso recursivo lo antes posible. Para ello, deberíamos arribar a particiones del conjunto de entrenamiento que sean totalmente puras, de la manera más rápida posible. Si tuviéramos una medida de la pureza de cada nodo, deberíamos elegir el atributo que produce los nodos hijos más puros.

Antes de analizar algunas medidas formales sobre la pureza (o impureza) de las particiones que resultan de la selección de un atributo, consideremos un conjunto de entrenamiento para el problema de aprender el concepto $JugarTenis$, que se muestra en la tabla 1. En este caso, se tiene un conjunto de atributos $\mathcal{A} = \{Estado, Temperatura, Humedad, Viento, JugarTenis\}$ y la tarea de aprendizaje

consiste en predecir el atributo objetivo $A_{\mathcal{O}} = JugarTenis$ que puede tener valores si o no, en base a los otros atributos de un día arbitrario, dados por el conjunto $\mathcal{A}^- \equiv \mathcal{A} - \{JugarTenis\}$. Los valores posibles para los atributos de $\mathcal{A}^-$ son:

$$
\begin{aligned}
V(Estado) &= \{soleado, nublado, lluvioso\} \\
V(Temperatura) &= \{caluroso, templado, fresco\} \\
V(Humedad) &= \{alta, normal\} \\
V(Viento) &= \{verdadero, falso\}
\end{aligned}
$$

Ejemplo	Atributos				Clase ($A_{\mathcal{O}}$)
	$Estado$	$Temperatura$	$Humedad$	$Viento$	$JugarTenis$
e_1	$soleado$	$caluroso$	$alta$	$falso$	no
e_2	$soleado$	$caluroso$	$alta$	$verdadero$	no
e_3	$nublado$	$caluroso$	$alta$	$falso$	si
e_4	$lluvioso$	$templado$	$alta$	$falso$	si
e_5	$lluvioso$	$fresco$	$normal$	$falso$	si
e_6	$lluvioso$	$fresco$	$normal$	$verdadero$	no
e_7	$nublado$	$fresco$	$normal$	$verdadero$	si
e_8	$soleado$	$templado$	$alta$	$falso$	no
e_9	$soleado$	$fresco$	$normal$	$falso$	si
e_{10}	$lluvioso$	$templado$	$normal$	$falso$	si
e_{11}	$soleado$	$templado$	$normal$	$verdadero$	si
e_{12}	$nublado$	$templado$	$alta$	$verdadero$	si
e_{13}	$nublado$	$caluroso$	$normal$	$falso$	si
e_{14}	$lluvioso$	$templado$	$alta$	$verdadero$	no

Tabla 1: Un conjunto de entrenamiento pequeño

El conjunto de entrenamiento consiste de 14 ejemplos, 9 de los cuales están rotulados como positivos ($JugarTenis = si$) y 5 como negativos ($JugarTenis = no$). Para analizar el grado de pureza en los conjuntos resultante de una partición, observemos en primer lugar en la figura 3, cual sería el efecto de particionar el conjunto de entrenamiento de la tabla 1 de acuerdo al atributo $Estado$.

En la tabla 1 podemos ver, que el conjunto de entrenamiento inicial $E = \{e_1, e_2, \ldots, e_{14}\}$ tiene 9 ejemplos positivos y 5 ejemplos negativos (que denotamos como $[9+, 5-]$) y que si E es particionado de acuerdo al atributo $Estado$, obtendremos 3 subconjuntos disjuntos $E_{soleado}$, $E_{nublado}$ y $E_{lluvioso}$, uno por cada valor del atributo $Estado$. Cada uno de estos subconjuntos E_v donde $v \in V(Estado)$, tendrá aquellos ejemplos de entrenamiento del conjunto E, que tengan en el atributo $Estado$ el valor v. Así por ejemplo, en la figura 3 podemos observar que $E_{soleado} = \{e_1, e_2, e_8, e_9, e_{11}\}$ y que en este subconjunto 2 ejemplos clasifican $JugarTenis$ como positivo (e_9 y e_{11}) y 3 ejemplos como negativo (e_1, e_2 y e_8).

Si uno deseara analizar en forma intuitiva cual es el grado de (im)pureza que resulta de la partición del conjunto E de acuerdo a los 4 atributos del conjunto $\mathcal{A}^-$, tendríamos una situación como la que se muestra en la figura 4. De acuerdo a estas particiones, ¿cuál sería a su criterio el mejor atributo para

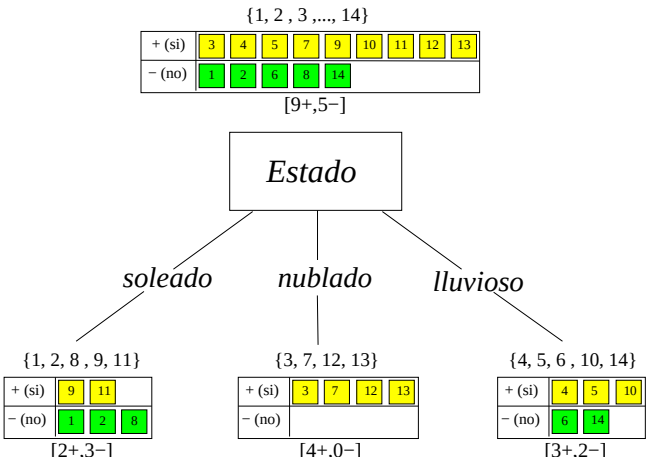
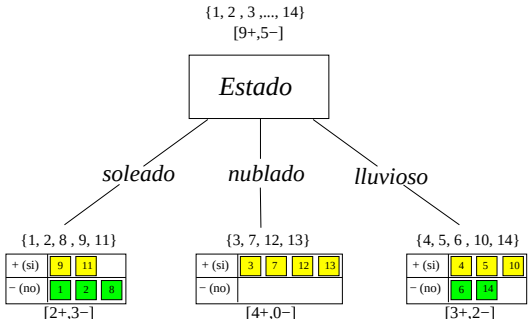
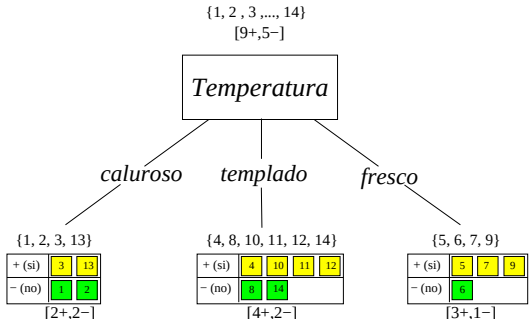


Figura 3: Partición del conjunto de entrenamiento de acuerdo al atributo $Estado$.

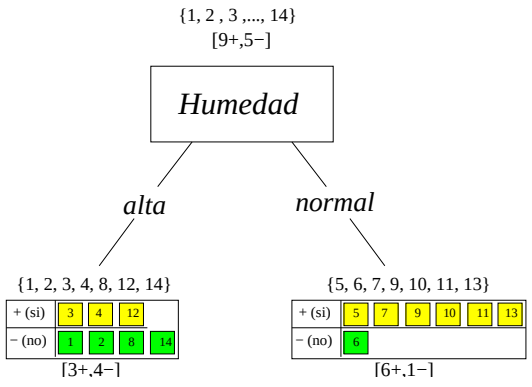
elegir como raíz del árbol?. Si el objetivo es buscar aquellos atributos que conducen a particiones más puras (homogéneas) el atributo $Estado$ parecería ser la mejor opción. Es la única alternativa para la cual un nodo hijo es completamente puro. El atributo $Humedad$ parecería ser la mejor segunda opción, ya que produce un nodo hijo más grande que es casi completamente puro.



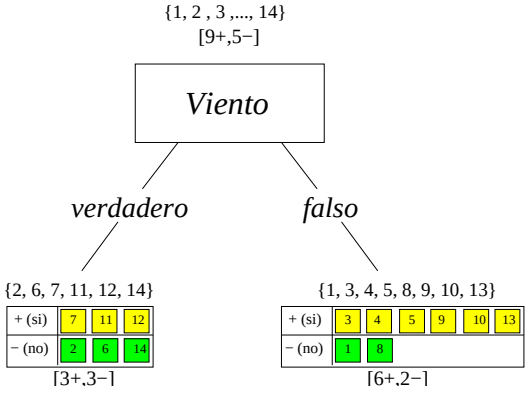
(a) Atributo $Estado$.



(b) Atributo $Temperatura$.



(c) Atributo $Humedad$.



(d) Atributo $Viento$.

Figura 4: Particiones de E de acuerdo a los distintos atributos.

Dado que es necesario llevar estas ideas intuitivas a la práctica, requeriremos una medida más precisa sobre la (im)pureza de un conjunto de ejemplos que pueda ser directamente traducida en un

programa de computación. La teoría de la información nos brinda tal herramienta mediante el concepto conocido como *entropía*.

La entropía puede ser considerada la *cantidad de información* contenida en el resultado de un experimento. Dicha información, dependerá sobre el *conocimiento previo* que se tiene sobre los resultados del experimento. Cuanto menos se conoce más información se obtiene (o más se aprende).

Si un experimento puede tener m resultados distintos $v_1, \ldots, v_m$ que pueden ocurrir con probabilidades $P(v_1), \ldots, P(v_m)$, la cantidad de información I que se obtiene al conocer el resultado real del experimento es:

$$I(P(v_1), \ldots, P(v_m)) \equiv \sum_{i=1}^{m} -P(v_i) \log_2 P(v_i) \tag{1}$$

Para ejemplificar esta idea, consideremos el experimento de arrojar una moneda, el cual tiene como resultados posible $cara$ y $ceca$. Si conocemos de antemano que la moneda fué alterada para que siempre caiga $cara$, la entropía (información) I del resultado del experimento será[2]:

$$I(P(cara), P(ceca)) = I(1, 0) = -1 \log_2 1 - 0 \log_2 0 = 0$$

Este resultado significa que, dado que ya sabemos que la moneda caerá $cara$, la información que obtengamos al conocer el resultado del experimento será nula. Si en cambio utilizamos una moneda totalmente balanceada, que produce cualquiera de los dos resultados en forma equiprobable, tendremos que:

$$I\left(\frac{1}{2}, \frac{1}{2}\right) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

Como podemos observar la entropía tiene su valor más bajo (0) cuando existe total certeza en el resultado del experimento, mientras que el mayor valor de entropía es alcanzado en el caso de mayor incertidumbre (eventos equiprobables). Entre estos dos valores extremos tendremos toda una serie de distribuciones de probabilidad válidas caracterizadas por tener una entropía baja cuando existen eventos altamente probables. Así por ejemplo, si la moneda es alterada para caer $cara$ en un $99\,\%$ de los casos, tendremos que $I(\frac{99}{100}, \frac{1}{100}) = 0{,}08$.

Si bien para este caso, con dos resultados posibles, los valores de entropía se encuentran en el intervalo $[0, 1]$, para el caso general con m resultados posibles y $m > 2$ se cumple que $0 \leq I \leq \log_2 m$.

Para llevar estas ideas al problema de determinar el grado de (im)pureza de un conjunto de entrenamiento E, observemos que este conjunto puede ser considerado una muestra del atributo objetivo $A_{\mathcal{O}}$. En este caso, las probabilidades de ocurrencia $P(v_i)$ de cada valor $v_i \in V(A_{\mathcal{O}})$ podrían ser estimadas como la proporción p_{v_i} de ejemplos en E que tienen a v_i como valor de $A_{\mathcal{O}}$:

$$p_{v_i} = \frac{n_{v_i}}{\sum_{v_j \in V(A_{\mathcal{O}})} n_{v_j}} \tag{2}$$

donde n_{v_k} es el número de ejemplos de entrenamiento en E que tienen a v_k como valor de $A_{\mathcal{O}}$. De esta manera, si denotamos al conjunto E como $[n_{v_1}, \ldots, n_{v_m}]$, o bien $[p_{v_1}, \ldots, p_{v_m}]$, la entropía de E respecto a la clasificación $A_{\mathcal{O}}$ puede ser calculada como:

$$I(E) = I([n_{v_1}, \ldots, n_{v_m}]) = I([p_{v_1}, \ldots, p_{v_m}]) = -\sum_{v_j \in V(A_{\mathcal{O}})} p_{v_j} \log_2 p_{v_j} \tag{3}$$

[2]En todos los cálculos involucrados con la entropía asumiremos que $0 \log 0 = 0$.

Para el caso particular de un atributo objetivo booleano, tendremos una colección E con ejemplos positivos y negativos del concepto objetivo, y la entropía de E relativa a esta clasificación booleana será:

$$I(E) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

donde $p_{\oplus}$ es la proporción de ejemplos positivos en E y $p_{\ominus}$ es la proporción de ejemplos negativos en E. La figura 5, muestra la forma de la función de entropía relativa a esta clasificación booleana, con $p_{\oplus}$ variando entre 0 y 1.

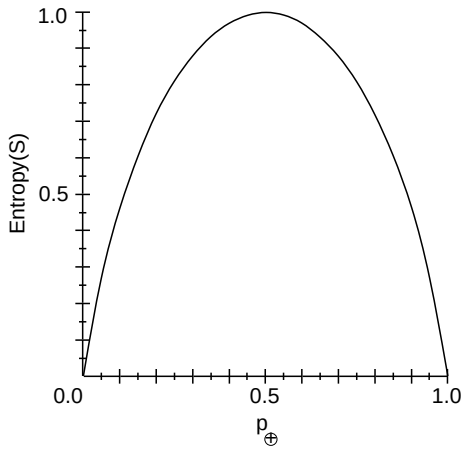


Figura 5: Función de entropía relativa a una clasificación booleana.

Continuando con el ejemplo anterior, si E es la colección de 14 ejemplos del concepto booleano $JugarTenis$ de la tabla 1, vemos que E que contiene 9 ejemplos positivos y 5 negativos, y que la entropía de E relativa a esta clasificación booleana es:

$$\begin{aligned} I(E) = I([9+, 5-]) &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0{,}940 \end{aligned}$$

Podemos notar que la mínima entropía es 0, cuando todos los miembros pertenecen a la misma clase, y la máxima es 1 cuando la colección contiene el mismo número de ejemplos positivos y negativos. Por lo tanto, la entropía puede ser utilizada como una medida de la (im)pureza de un conjunto de datos, tomando un valor 0 cuando los datos son totalmente puros (pertenecen a la misma clase) y el valor de máxima impureza cuando existe igual proporción de ejemplos de cada una de las clases del atributo objetivo.

Observar que de acuerdo al concepto de entropía, el algoritmo de aprendizaje de árboles de decisión, tiene como punto de parada en el paso 2 de la figura 2 el caso en que $I(E) = 0$ (máxima pureza). Sin embargo, normalmente los ejemplos no son puros ($I(E) \neq 0$) y será necesario analizar la (im)pureza de las particiones que resultan de considerar los valores posibles de algún atributo $A \in Atributos$. A esta cantidad se la denomina la *información residual* de E respecto a A y la denotaremos $I_{res}(E, A)$.

La *información residual* $I_{res}(E, A)$ se obtiene tomando como base el concepto de *entropía condi-*

cional promedio $H(Y|X)$ [3] que se define como:

$$\begin{aligned} H(Y|X) &\equiv \sum_{x_i} P(x_i) \cdot H(Y|x_i) \\ &= \sum_{x_i} P(x_i)(-\sum_{y_i} P(y_i|x_i) \log_2 P(y_i|x_i)) \end{aligned}$$

Llevando estas ideas al contexto de un conjunto de entrenamiento E y un atributo arbitrario A, vemos que $I_{res}(E, A)$ puede ser definida como una estimación de $H(A_{\mathcal{O}}|A)$ que se obtiene sumando las entropías de los subconjuntos resultantes de particionar E de acuerdo a A, ponderadas por la probabilidad de ocurrencia de los valores de A:

$$\begin{aligned} I_{res}(E, A) &\equiv \sum_{v \in V(A)} P(v) \cdot (-\sum_{c \in V(A_{\mathcal{O}})} P(c|v) \log_2 P(c|v)) \\ &= \sum_{v \in V(A)} P(v) \cdot I(E_v) \\ &= \sum_{v \in V(A)} \frac{|E_v|}{|E|} \cdot I(E_v) \end{aligned}$$

La información residual respecto a un atributo A es la información que aún necesitaremos para clasificar una instancia *después* de testear el atributo A. Por lo tanto, el paso 4 del algoritmo de aprendizaje de la figura 2 podría limitarse a elegir aquel atributo A con menor información residual $I_{res}(E, A)$.

Otra alternativa consiste en considerar la *ganancia de información* $G(E, A)$[4] que se obtiene al testear el atributo A, y que se calcula como la diferencia entre el requerimiento de información original y la información requerida luego de testar el atributo:

$$\begin{aligned} G(E, A) &\equiv I(E) - I_{res}(E, A) \\ &\equiv I(E) - \sum_{v \in V(A)} \frac{|E_v|}{|E|} \cdot I(E_v) \end{aligned}$$

La ganancia de información es simplemente la reducción esperada en la entropía causada al particionar los ejemplos de acuerdo a un atributo. Por lo tanto, en el paso 4 del algoritmo ID3, el atributo seleccionado debería ser aquel con *mayor ganancia de información*. Para el caso del aprendizaje del concepto $JugarTenis$, si tomamos por ejemplo el atributo $Viento$ tendremos que

$$\begin{aligned} V(Viento) &= \{falso, verdadero\} \\ E &= [9+, 5-] \\ E_{falso} &\leftarrow [6+, 2-] \\ E_{verdadero} &\leftarrow [3+, 3-] \end{aligned}$$

[3] La notación utilizada en este caso, es la usual en teoría de la información.

[4] Usando notación de teoría de la información, la ganancia de información es $G(Y|X) = H(Y) - H(Y|X)$.

y la ganancia de información para este atributo puede ser calculada como:

$$\begin{aligned} G(E, Viento) &= I(E) - \sum_{v \in \{falso, verdadero\}} \frac{|E_v|}{|E|} I(E_v) \\ &= I(E) - (8/14) I(E_{falso}) - (6/14) I(E_{verdadero}) \\ &= 0{,}940 - (8/14) 0{,}811 - (6/14) 1{,}00 \\ &= 0{,}048 \end{aligned}$$

Si calculamos la ganancia de información para todos los atributos, obtendremos los siguientes resultados:

$$\begin{aligned} G(E, Estado) &= 0{,}246 \\ G(E, Humedad) &= 0{,}151 \\ G(E, Viento) &= 0{,}048 \\ G(E, Temperatura) &= 0{,}029 \end{aligned}$$

y por lo tanto el algoritmo ID3 seleccionará $Estado$ creando bajo este nodo una rama por cada posible valor de este atributo. El árbol de decisión parcial resultante en este caso, ya fue mostrado en la figura 3, junto con los ejemplos de entrenamiento que se asignan a cada nuevo nodo del árbol y sobre los cuales ID3 será aplicado recursivamente.

Observar que todos los ejemplos para los cuales $Estado = nublado$ son ejemplos positivos de $JugarTenis$, y por lo tanto este nodo se transforma en un nodo hoja con la clasificación $JugarTenis = si$. Por el contrario, los descendientes que corresponden a $Estado = soleado$ y $Estado = lluvioso$ tienen aún entropía distinta de 0, y el árbol deberá ser elaborado bajo estos nodos. El proceso de seleccionar un nuevo atributo y particionar los ejemplos de entrenamiento es repetido ahora para cada nodo descendiente no-terminal, usando esta vez sólo los ejemplos de entrenamiento asociados con este nodo. Los atributos que han sido incorporados más alto en el árbol son excluídos y por consiguiente, cualquier atributo puede aparecer a lo sumo una vez en cualquier paso del árbol. El proceso continúa para cada nuevo nodo hoja, hasta que alguna de las siguientes 2 condiciones es alcanzada:

1. Todo atributo ya ha sido incluido en este paso a través del árbol.
2. Todos los ejemplos de entrenamiento asociados con este nodo hoja tienen el mismo valor de atributo objetivo (entropía = 0).

El árbol de decisión final aprendido por ID3, desde los 14 ejemplos de entrenamiento para el ejemplo de $JugarTenis$, es el que se mostró en la figura 1.

5. Búsqueda en el espacio de hipótesis en el aprendizaje de árboles de decisión

El espacio de hipótesis en el que busca ID3 es el conjunto de todos los árboles de decisión. ID3 realiza una búsqueda Hill-Climbing a través de este espacio de hipótesis, comenzando con el árbol vacío

y considerando en forma progresiva hipótesis más elaboradas, en busca de un árbol de decisión que clasifica correctamente los datos de entrenamiento. La función de evaluación que guía esta búsqueda Hill-Climbing es la medida de ganancia de información. Una visión intuitiva de la forma en que ID3 realiza esta búsqueda se muestra en la figura 6.

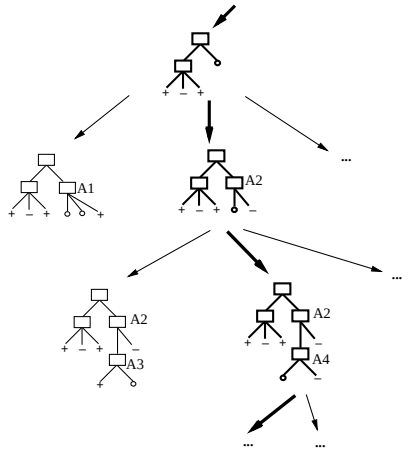


Figura 6: Búsqueda en el espacio de hipótesis de ID3.

Visualizando ID3, en términos de su espacio de búsqueda y estrategia de búsqueda, podemos realizar las siguientes observaciones en cuanto a sus capacidades y limitaciones:

1. El espacio de hipótesis que recorre ID3 (todos los árboles de decisión) es un espacio completo de funciones discretas finitas (toda función de este tipo puede ser representada por un árbol de decisión).

2. ID3 mantiene sólo una hipótesis a medida que busca a través del espacio de árboles de decisión.

3. ID3 en su forma pura no realiza back-tracking en su búsqueda (peligro de todas las búsquedas Hill-Climbing de converger a un óptimo local).

4. ID3 usa todos los ejemplos de entrenamiento en cada paso de la búsqueda para hacer decisiones basadas en estadísticas. Por este motivo, es menos sensitivo a errores en los datos de entrenamiento, que aquellos métodos que trabajan con ejemplos provistos en forma incremental.

El sesgo inductivo[5] de ID3 podría ser caracterizado en primera instancia como una *preferencia por árboles de decisión más cortos sobre árboles complejos*. Notar que este sesgo también sería exhibido (y en forma más concluyente) por un algoritmo *breadth first* que comienza con el árbol vacío y considera progresivamente *todos* los árboles de profundidad 1, luego los de profundidad 2 y así sucesivamente hasta encontrar un árbol de decisión consistente con los ejemplos de entrenamiento. En este caso, el árbol de decisión retornado por este algoritmo que denominaremos BFS-ID3 (*breadth-first search ID3*) será el árbol de decisión más pequeño (con la menor cantidad posible de nodos) consistente con los ejemplos de entrenamiento. En este sentido, ID3 puede ser considerado una aproximación eficiente de BFS-ID3 que usa una búsqueda heurística greedy para intentar encontrar el árbol más corto sin realizar

[5]El sesgo inductivo es el conjunto de asumciones que, conjuntamente con los datos de entrenamiento, justifican deductivamente las clasificaciones asignadas por el aprendiz a las instancias futuras

una búsqueda breadth-first exhaustiva sobre el espacio de hipótesis. Sin embargo, dado que ID3 usa la heurística de ganancia de información y una estrategia hill climbing, su sesgo es algo más complejo que el de BFD-ID3 ya que puede incluso, no siempre encontrar el árbol de decisión consistente más corto. Por lo tanto, una aproximación más cercana al sesgo inductivo de ID3 podría ser caracterizado como:

> *Los árboles más cortos son preferidos sobre los más largos. A su vez, aquellos árboles que ubican atributos con alta ganancia de información más cerca de la raíz son preferidos a aquellos que no lo hacen.*

ID3 impone esencialmente un *sesgo de búsqueda* en lugar de un *sesgo de restricción* (o *lenguaje*). Este sesgo que favorece los árboles de decisión más cortos está fundado en el principio conocido como "Occam's razor" que establece: *preferir la hipótesis más simple que se ajusta a los datos*. Este principio puede ser fundamentado en el hecho de que, dado que existe una menor cantidad de hipótesis pequeñas que complejas, es menos probable que uno encuentre una hipótesis corta que coincida en forma accidental con los datos.

6. Medidas alternativas para seleccionar atributos

Existe una tendencia natural en la medida de ganancia de información de favorecer a los atributos con *muchos valores* sobre aquellos con pocos valores. Tomemos el caso extremo de un atributo que tiene un valor diferente para cada instancia en el conjunto de datos, como puede ser por ejemplo un *código de identificación*. Para ejemplificar esta situación, supongamos que la primera columna de la tabla 1 utilizada para identificar los ejemplos, representa en realidad los valores de un nuevo atributo que denominaremos ID. En este caso, si analizamos la partición del conjunto E que resulta de considerar este atributo obtendremos el resultado que se muestra en la figura 7. Es sencillo verificar en este caso que $I(E_{e_k}) = 0$ para todo $e_k \in V(ID)$ y por lo tanto, la información residual con respecto a ID será $I_{res}(E, ID) = 0$. De acuerdo a esta cantidad, la ganancia de información obtenida con ID será $G(E, ID) = I(E) - I_{res}(E, ID) = 0{,}940$, que supera a la ganancia de información del resto de los atributos y por lo tanto sería seleccionado por ID3 como atributo raíz.

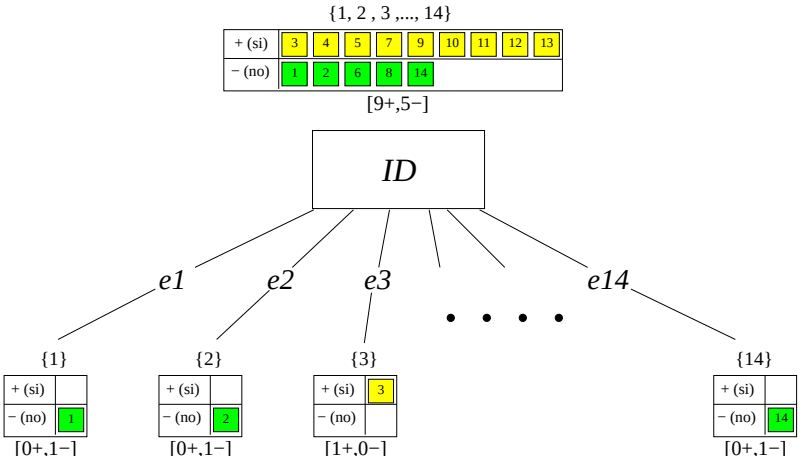


Figura 7: Partición del conjunto de entrenamiento de acuerdo al atributo ID.

El problema con seleccionar un atributo como ID es que si bien predice perfectamente el atributo objetivo sobre los datos de entrenamiento, no es bueno para predecir la clase de las instancias *no conocidas* y no dice nada sobre la estructura de decisión que será utilizada para generalizar a otros casos. Si bien un atributo del tipo ID constituye un caso extremo, este problema es extensible a otros atributos con muchos valores (como por ejemplo *fechas*) que suelen exhibir características similares. Para solucionar este problema, un enfoque consiste en dejar de lado las medidas basadas en la entropía y ganancia de la información y utilizar otro tipo de medida evaluatoria de los atributos, como por ejemplo el *índice de Ginni*. Otro enfoque, se basa en tratar de ajustar la medida de ganancia de información de manera tal de penalizar aquellos atributos que tengan un número muy grande de valores posibles. Esta idea dá origen al concepto de *radio de ganancia* que será explicado a continuación.

Para entender los fundamentos del radio de ganancia observemos qué sucedería si uno definiera la entropía de un conjunto de datos, no en función del atributo objetivo $A_{\mathcal{O}}$ sino en función de un atributo arbitrario A. En este caso, la entropía de un conjunto de entrenamiento E respecto al atributo A, que denotaremos $SI(E, A)$, podría ser obtenida reescribiendo la ecuación 3 de la entropía de la siguiente manera:

$$\begin{aligned} SI(E, A) &= -\sum_{v \in V(A)} p_v \log_2 p_v \\ &= -\sum_{v \in V(A)} \frac{|E_v|}{|E|} \log_2 \frac{|E_v|}{|E|} \end{aligned} \tag{4}$$

Cuando la entropía es medida con respecto a un atributo A, $SI(E, A)$ será alta cuando el atributo particione E en una gran cantidad de subconjuntos pequeños, es decir, cuando el atributo particiona los ejemplos de manera amplia y uniforme, generando una gran cantidad de nodos hijos. Es por este motivo que $SI(E, A)$ suele ser referenciada como la *información de particionado* (*split information*). Esta información puede ser utilizada como un término de penalización para los atributos con muchos valores, dando origen a la medida denominada *radio de ganancia* de un atributo A, que denotaremos $GR(E, A)$ y que se define como:

$$GR(E, A) \equiv \frac{G(E, A)}{SI(E, A)} \tag{5}$$

Es sencillo verificar en este caso que $SI(E, ID) = 3{,}807$, mientras que un atributo como $Estado$ tendrá un valor de información de particionado de $SI(E, Estado) = 1{,}577$ y por lo tanto la penalización que reciba el atributo $Estado$ en el cálculo del radio de ganancia será mucho menor que cuando se utiliza el atributo ID. Los valores de radio de ganancia de todos los atributos son los siguientes:

$$\begin{aligned} GR(E, ID) &= 0{,}940/3{,}807 = 0{,}246 \\ GR(E, Estado) &= 0{,}246/1{,}577 = 0{,}156 \\ GR(E, Humedad) &= 0{,}151/1 = 0{,}151 \\ GR(E, Viento) &= 0{,}048/0{,}985 = 0{,}049 \\ GR(E, Temperatura) &= 0{,}029/1{,}362 = 0{,}021 \end{aligned}$$

Podemos observar que si bien el atributo hipotético ID sería aún preferido al atributo $Estado$, su ventaja se ha reducido significativamente con respecto a la relación cercana a 4 a 1 que se obtenía con la ganancia de información. Este aspecto es el motivo por el cual, en las implementaciones prácticas, es común utilizar algún test *ad hoc* para prevenir la partición de los ejemplos utilizando estos atributos inútiles. Por otra parte, es posible observar que si sacamos de consideración el atributo ID, el atributo $Estado$ seguirá siendo la alternativa preferida de acuerdo al radio de ganancia con una leve ventaja sobre el atributo $Humedad$. Sin embargo, la diferencia en este caso es menor que cuando se utilizaba la ganancia de información. ¿Cuál es el motivo de ésto?.

Un aspecto negativo del uso del radio de ganancia es que en algunas situaciones puede conducir a que se prefiera un atributo por el único motivo de que su información intrínseca es mucho menor que la de los otros atributos. Una forma usual de solucionar este problema es seleccionar el atributo que maximiza el radio de ganancia, en la medida que la ganancia de información para este atributo sea al menos tan alta como la ganancia de información promedio de todos los atributos considerados.

7. Atributos numéricos

Hasta el momento sólo hemos considerado atributos nominales con un conjunto discreto de valores. Sin embargo, la mayoría de los conjuntos de datos reales contienen atributos numéricos continuos. Este tipo de atributos pueden ser fácilmente incorporados al esquema previo, mediante la definición dinámica de nuevos atributos que particionan los atributos continuos en un conjunto discreto de intervalos. La idea en este caso, es que un atributo numérico A puede ser particionado en forma binaria, creando dinámicamente un nuevo atributo booleano A_c que es verdadero si $A < c$ y falso en otro caso. La única pregunta que resta responder es cual es el mejor valor para el umbral (*threshold*) c. Considerando que el atributo A_c será comparado con otros atributos de acuerdo a su ganancia de información, es obvio que el valor de c debería ser aquel entre todos los umbrales posibles, que maximiza la ganancia de información.

Si uno ordena los ejemplos de entrenamiento de acuerdo a un atributo numérico A, es posible identificar los ejemplos adyacentes en que se producen los cambios en la clasificación objetivo. Un conjunto de umbrales candidatos en este caso, podría estar dado por los valores intermedios del atributo A en que dichos cambios se producen. Esta elección de los umbrales candidatos no es arbitraria. Se ha demostrado que el valor de c que maximiza la ganancia de información siempre estará ubicado en alguno de estos puntos. Por lo tanto, una aproximación para encontrar el umbral óptimo de c consistirá en, dados los posibles puntos intermedios para el umbral, seleccionar aquel que produce la mayor ganancia de información.

Para apreciar estas ideas en un ejemplo concreto, asumamos que el atributo $Temperatura$, a diferencia del ejemplo de las secciones previas, no es un atributo nominal, sino un atributo numérico (entero). Consideremos además, que los valores de $Temperatura$ que se han observado en los ejemplos de entrenamiento asociados con un nodo particular son los mostrados en la tabla 2 junto con los valores correspondientes para el atributo objetivo $JugarTenis$.

$Temperatura$:	40	48	60	72	80	90
$JugarTenis$:	no	no	si	si	si	no

Tabla 2: Un conjunto de entrenamiento pequeño

En este ejemplo, existen dos umbrales candidatos indicados por las dos líneas verticales sobre la tabla, que corresponden a los valores de $Temperatura$ donde el valor de $JugarTenis$ cambia: $(48 + 60)/2$, y $(80 + 90)/2$. La ganancia de información puede ser calculada para cada uno de los atributos candidatos, $Temperatura_{<54}$ y $Temperatura_{<85}$, y el mejor de los dos puede ser seleccionado ($Temperatura_{<54}$).

8. Ejemplos con valores desconocidos de los atributos

En la etapa previa a la aplicación de un algoritmo de aprendizaje, se suele realizar alguna forma de "limpieza" de los datos de entrenamiento para solucionar problemas frecuentes como por ejemplo la existencia de ejemplos de entrenamiento que tienen valores faltantes para uno o más de sus atributos. La solución en este caso, puede consistir en adoptar criterios muy drásticos, como cuando se eliminan el/los atributo/s (columnas) afectados o bien se eliminan las instancias completas (filas) que tienen valores faltantes. Otra alternativa es realizar alguna forma de "reparación" de los datos, completando los mismos en forma *manual* o incluso mediante técnicas automáticas más sofisticadas que incluyen la predicción de los valores faltantes mediante un algoritmo de aprendizaje automático. Por otra parte, si el hecho de que la falta del valor para un atributo es significativo, también se puede considerar al valor faltante como otro valor posible del atributo.

Estas técnicas previas a la aplicación del algoritmo de aprendizaje automático, se suelen basar en criterios muy generales y en muchos casos se opta por dejar los datos de entrenamiento incompletos y asumir que el algoritmo de aprendizaje manejará este problema en forma robusta. Para el caso particular del aprendizaje de árboles de decisión, el problema se nos presenta cuando existe un ejemplo de entrenamiento $e = \langle x, c(x) \rangle \in E$ que tiene un valor desconocido para el atributo A y necesitamos estimar $x(A)$ para poder calcular $G(E, A)$ en un nodo arbitrario n.

Una estrategia para solucionar este problema consiste en considerar que $x(A)$ es el valor *más común* del atributo A entre los ejemplos de E en el nodo n. Alternativamente, podemos tomar como $x(A)$ al valor más común del atributo A entre los ejemplos de E en el nodo n, *que tienen la clasificación* $c(x)$. En cualquiera de estos dos casos, el ejemplo de entrenamiento elaborado para asumir este valor de $x(A)$ podrá ser utilizado *directamente* por un algoritmo de aprendizaje de árboles de decisión.

Un procedimiento más complejo (utilizado en el algoritmo C4.5) consiste en "particionar" (conceptualmente) al ejemplo e en tantas piezas e_i, como valores $v_i \in V(A)$ existan. Cada pieza e_i es ponderada con una probabilidad p_i que puede ser estimada en base a la frecuencia observada del valor v_i en los ejemplos de entrenamiento en el nodo n. Posteriormente, el valor p_i puede ser utilizado en todos los cálculos para determinar la ganancia del atributo A [6] y cada fracción de la instancia e es distribuída por la rama correspondiente al valor del atributo A.

9. Atributos con diferentes costos

En ciertos dominios algunos atributos pueden ser *más caros* que otros. Pensemos por ejemplo en el diagnóstico de enfermedades en un hospital, donde los datos correspondientes a un paciente pueden incluir el análisis de sangre, radiografías, ecografías, electrocardiogramas, control de temperatura, etc. Indudablemente, en estos casos la obtención de los valores para estos atributos podrá tener un costo asociado que puede ser medido en dinero, tiempo, bienestar del paciente o riesgo asociado. En este

[6] Conceptualmente se puede visualizar a las instancias normales como ponderadas con una probabilidad de 1.

tipo de situaciones, es conveniente usar los atributos más baratos y recurrir a los caros sólo cuando se necesitan para producir predicciones confiables.

Para lograr que el algoritmo de aprendizaje tome en cuenta estos aspectos, una alternativa frecuentemente utilizada consiste en modificar la medida de ganancia de información de manera adecuada para que refleje la importancia de los costos de los atributos. La filosofía en este caso es similar a la adoptada para el cálculo del radio de ganancia donde se penalizaba a los atributos con muchos valores.

Una primera aproximación podría consistir en reemplazar la ganancia por una medida como:

$$\frac{G(E,A)}{Costo(A)}$$

de manera tal que los atributos de costo más bajo sean preferidos. Mientras esta medida no garantiza encontrar un árbol de decisión sensitivo al costo que sea óptimo, produce un sesgo en la búsqueda en favor de los atributos más baratos.

Distintas variantes del esquema anterior han sido propuestas, que difieren esencialmente en la forma de ponderar la ganancia y el costo de los atributos, un factor que suele ser muy dependiente del dominio de aplicación. Así por ejemplo, la siguiente medida:

$$\frac{G^2(E,A)}{Costo(A)}$$

ha sido utilizada en un robot para clasificar distintos objetos de acuerdo a cómo pueden ser sujetados para levantarlos. En este caso, el costo es medido por la cantidad de segundos que insume el posicionamiento y operación de los sensores utilizados para percibir el ambiente.

En el ámbito del diagnostico médico, se ha utilizado la medida:

$$\frac{2^{G(E,A)}-1}{(Costo(A)+1)^w}$$

donde los atributos corresponden a distintos síntomas y tests de laboratorio que tienen diferentes costos. En este caso, el parámetro $w \in [0,1]$ determina la importancia relativa del costo versus la ganancia de información.