UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

---

Estructura de Datos
Práctica Final : ¿Quién es Quién ?

Autores :
Alejandro Jerónimo Fuentes (B1)
Elvira Castillo Fernández (D3)

Fecha : 8 de enero de 2018

Construcción del árbol

Construcción básica

En primer lugar construiremos el árbol de la forma más sencilla posible. La forma más sencilla consiste en recorrer el árbol por niveles y en cada nivel formular una pregunta e ir descartando los personajes. Para ello, se han creado las siguientes funciones:

- `void crear_arbol()` ⟶ Esta función inicializa el árbol del juego con un primer nodo (nodo raíz) que corresponde al primer atributo del tablero que será la primera pregunta. Dentro de la función se realiza una llamada a `construccion_basica()`
- `void construccion_basica(vector<int> descartes, bintree<Pregunta>::node nodo,int nivel)`

  La función anterior va creando en cada nivel del árbol las distintas preguntas o personajes según los personajes que aún quedan por descartar. Para saber los personajes que quedan usamos un vector de enteros llamado `descartes`. El resto de argumentos consisten en un objeto de tipo nodo que será donde se introduzcan las preguntas o los personajes a la izquierda o a la derecha del nodo y el argumento `nivel` indica el nivel del árbol en el que nos encontramos.

El algoritmo utilizado para la construcción básica se describe a continuación:

En primer lugar creamos dos vectores de descartes, uno para los personajes que cumplen el atributo en la columna del tablero (se situarán en la rama izquierda) y otro para los personajes que no lo cumplen (rama derecha). Estos vectores se llenan recorriendo la columna del tablero correspondiente al nivel actual del árbol.

Una vez descartados los personajes se pueden dar tres casos:

1. Si no quedan personajes en el tablero entonces no se hace nada.
2. Si sólo queda un personaje en el vector de descartes entonces es un nodo hoja y lo insertamos donde corresponda (rama izquierda o derecha)
3. Finalmente si el vector de descartes tiene más de un elemento entonces estaremos seguros de que no es un personaje y que aún quedan por descartar. Por tanto formulamos la siguiente pregunta y llamamos recursivamente `construccion_Basica` para seguir profundizando en el árbol.

Algoritmo 1 Construcción básica del árbol

Entrada: descartes:[] **int**, nodo: **Nodo**, nivel: **int**

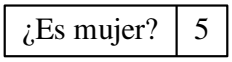
```
1: izda:[] int
2: dcha:[] int
3: personaje: Pregunta
4: pregunta: Pregunta
5:
6: izda := descartarPersonajesIzda()
7: dcha := descartarPersonajesDcha()
8:
9: si izda.tamaño() == 0 entonces
10:
11: fin si
12: si izda.tamaño() == 1 entonces
13:    personaje := crearPersonaje()
14:    nodo.insertarIzquierda(personaje)
15: fin si
16: si izda.tamaño() > 1 entonces
17:    pregunta := crearPregunta()
18:    nodo.insertarIzquierda(pregunta)
19:    construccionBasica(izda,nodo.hijoIzda,nivel+1)
20: fin si
21:
22:
23: si dcha.tamaño() == 0 entonces
24:
25: fin si
26: si dcha.tamaño() == 1 entonces
27:    personaje := crearPersonaje()
28:    nodo.insertarDerecha(personaje)
29: fin si
30: si dcha.tamaño() > 1 entonces
31:    pregunta := crearPregunta()
32:    nodo.insertarDerecha(pregunta)
33:    construccionBasica(dcha,nodo.hijoDcha,nivel+1)
34: fin si
```

Ejemplo

Por ejemplo, si tenemos el siguiente tablero:

Mujer	Ojos marrones	Pelo castaño	Tiene gafas	Nombre personaje
0	1	1	1	Ernesto
1	0	0	0	Ana
0	0	1	1	Juan
0	1	0	0	Antonio
1	1	1	0	Pilar

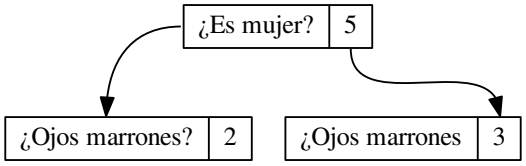
En primer lugar tenemos que insertar la raíz en el árbol. La raíz será el primer elemento del vector de atributos, en este caso la primera pregunta será si es mujer. El número de personajes será 5 ya que al principio no se ha descartado nada.



Una vez teniendo la raíz llamamos por primera vez a la función `construccion_basica` y le pasamos el vector de descartes que será el vector de personajes, el nodo raíz del árbol y el nivel 0. Dentro de la función se crean los vectores `descartes_izda` y `descartes_dcha` que tendrán los personajes que cumplen el atributo mujer (izda) y los que no lo cumplen (dcha). El contenido de los vectores es:

- `descartes_izda`: 1 4 (Ana y Pilar)
- `descartes_dcha`: 0 2 3 (Ernesto,Juan y Antonio)

Como el tamaño de los vectores es mayor que 1 entonces no tenemos el personaje y hay que seguir preguntando. Para ello creamos a la izquierda y a la derecha de la raíz los nuevos nodos que tendrán la nueva pregunta y los personajes descartados y llamamos recursivamente a la función.

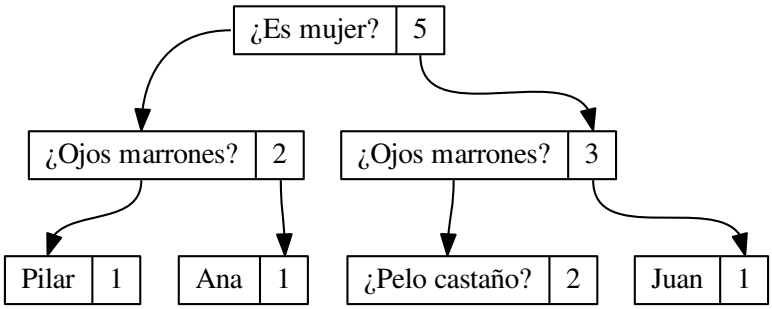


En el caso del hijo a la izquierda los nuevos vectores de descartes se calcularían cual de los dos personajes que nos quedan cumplen o no el atributo de los ojos marrones. Nos queda:

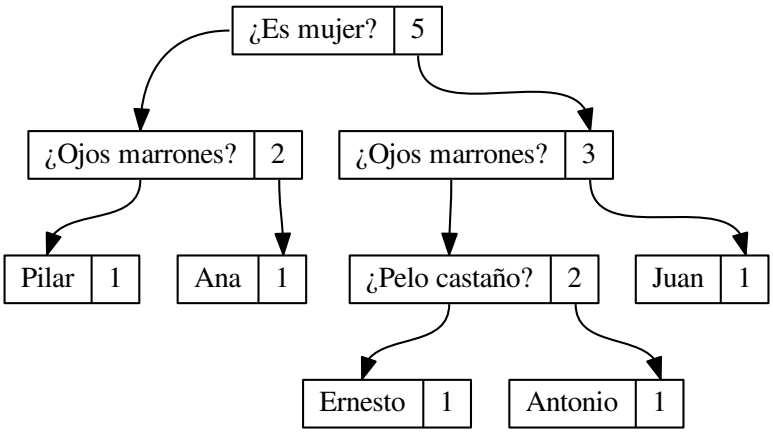
- `descartes_izda`: 4 (Pilar)
- `descartes_dcha`: 1 (Ana)

Como los tamaños de los vectores es igual a 1 en los dos casos, hemos llegado a un nodo hoja y lo insertamos.

En el caso del hijo a la derecha aún nos quedan personaje por descartar y el árbol nos quedaría:



Finalmente habría que realizar una última llamada a la pregunta Pelo castaño para obtener el árbol completo.



Mejora

Algoritmo ID3

La solución anterior construye un árbol en el que las preguntas se hacen siempre en el mismo orden en el que se metieron en el vector de preguntas y también nos interesa que dado cualquier tablero obtener un árbol equilibrado para hacer el menor número de preguntas posible.

Para ello podemos usar técnicas de la Inteligencia Artificial y la Teoría de la Información sobre árboles de decisión. Un árbol de decisión es aquel en el que los nodos interiores son preguntas y los nodos hoja son las decisiones que podemos tomar. Los árboles de decisión se basan en la experiencia para tomar las decisiones es por ello que se usan a través de ejemplos, uno de los principios del aprendizaje inductivo.

El algoritmo que se ha implementado como mejora de la construcción básica es ID3. ID3 elige en cada momento el mejor atributo y trata de obtener el árbol más óptimo posible pero sin garantizarlo.

Esto último es una aplicación directa de la técnica voraz (Greedy). Los algoritmos voraces seleccionan el elemento más prometedor para llegar a una solución rápidamente pero no siempre es la solución más óptima.
ID3 tiene los inconvenientes de que solo es aplicable a problemas de clasificación que tengan un número bajo de atributos.

Para obtener el mejor atributo aplicamos una heurística (criterio para seleccionar el mejor atributo) empleando los conceptos de entropía y ganancia:

- **Entropía:** Mide la incertidumbre de un determinado sistema.Es decir, representa la probibilidad de que ocurra cada uno de los posibles resultados. Si tenemos una entropía baja estaremos más seguros de que decisión tomar mientras que si tenemos una muy alta no podremos estar seguros de la decisión y habrá que seguir preguntando.

$$Entropia(S) = \sum_{i=1}^{n} -p_i log_2\,(p_i)$$

- **Ganancia:** mide lo que se gana en información en un determinado momento a partir de la entropía. El atributo que tenga mayor ganancia será el que se seleccione a continuación. Consiste en restarle a la entroía del sistema la entropía del sistema con el atributo.

$$Ganancia(Atributo) = Entropia(S) - Entropia(S, Atributo)$$

Elementos del algoritmo

- **Atributos:** Son los factores que influencian la clasificación o la decisión. En nuestro ejemplo ojos marrones, pelo castaño etc...
- **Clase:** Son los posibles valores de solución. En nuestro ejemplo todos los posibles personajes.
- **Ejemplo:** Es el conjunto de combinaciones de los atributos. En nuestro ejemplo todos los posibles valores de los atributos en cada fila.

Este algoritmo lo implementamos de manera recursiva con un vector de atributos. En cada paso se elige el mejor atributo y si la entropía en algún momento vale 0, estaremos ante un nodo hoja. Posteriormente se elimina el atributo del vector de atributos ya que no podemos volver a considerarlo.

Ejemplo

Retomando el ejemplo de la sección anterior vamos a realizar primero los cálculos para elegir nuestro primer atributo que será la raíz del árbol.

En primer lugar calculamos la entropía total. Para ello nos fijamos en los personajes que tenemos y como cada personaje es único el número de clases va a ser igual al número de personajes.

$$Entropia(S) = \sum_{i=1}^{5} -\frac{1}{5}\log_2\left(\frac{1}{5}\right) = 2{,}32$$

A continuación, calculamos la entropía en cada atributo y la restamos a la entropía del total:

Mujer (2 personajes lo cumplen y 3 no)

$$Entropia(2,3) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0{,}97$$

Ojos marrones (3 personajes lo cumplen y 2 no)

$$Entropia(3,2) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0{,}97$$

Pelo castaño (3 personajes lo cumplen y 2 no)

$$Entropia(3,2) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0{,}97$$

Tiene gafas (2 personajes lo cumplen y 3 no)

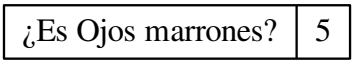
$$Entropia(2,3) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0{,}97$$

Como en este ejemplo tenemos la misma entropía para cada atributo la ganancia para todos los atributos es la misma y da igual elegir un atributo u otro.

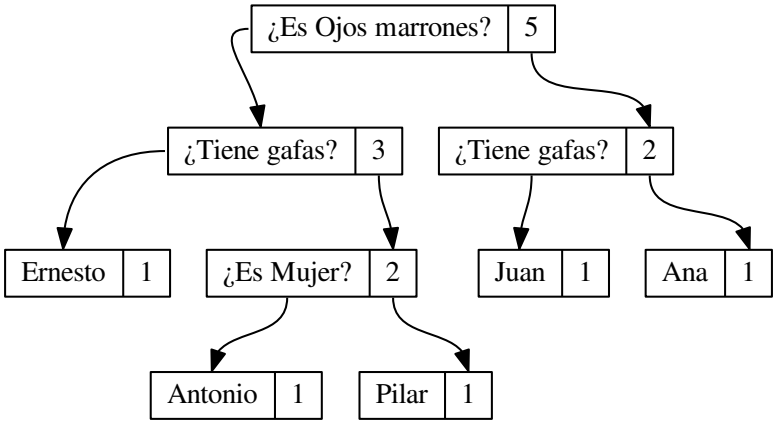
Por tanto:

$$Ganancia(Atributos) = 2{,}32 - 0{,}97 = 1{,}34$$

Elegimos como raíz cualquiera de los atributos. Por ejemplo ojos marrones:



Aplicando el algoritmo recursivamente obtenemos el árbol:



Conclusión

Observamos que con la aplicación de este algoritmo y los tableros facilitados para realizar la práctica no se obtienen grandes mejoras en la profundidad promedio. De media se hacen el mismo número de preguntas cuando jugamos, para llegar a la solución.

Según nuestra investigación, el algoritmo ID3 se emplea para el análisis de grandes volúmenes de datos (Data Science).

Quizás, no hemos obtenido resultados sustanciales de mejora, debido a la cantidad de datos examinada y a que el algoritmo de construcción básica junto con la eliminación de los nodos redundantes ya hace que el árbol esté prácticamente equilibrado. Con nuestro algoritmo pretendíamos elegir el atributo más prometedor y que el árbol estuviera equilibrado.

Por tanto la conclusión final es que mientras el árbol esté equilibrado y se hayan eliminado los nodos redundantes, se van a realizar en media, igual número de preguntas. Independientemente si usamos el algoritmo id3 o la construcción básica.