Aprendizaje de Árboles de Decisión Basados en Distancias*

V. Estruch C. Ferri J. Hernández-Orallo M.J. Ramírez-Quintana

Departamento de Sistemas Informáticos y Computación-DSIC
Universidad Politécnica de Valencia, C. de Vera s/n, 46022 Valencia, España.
{vestruch, cferri, jorallo, mramirez}@dsic.upv.es

Resumen Recientemente, se han propuesto varias aproximaciones para abordar el problema del aprendizaje sobre ejemplos estructurados o datos multi-relacionales. Sin embargo, muchas de ellas llevan a cabo su propósito sin retornar ningún modelo que represente el comportamiento en la clasificación. En este trabajo proponemos un nuevo algoritmo para la inferencia de árboles de decisión capaz de abordar problemas de clasificación en escenarios con datos complejos (estructurados o multi-relacionales). Dicho método recurre al concepto de distancia o de similitud para efectuar las particiones. Cada ejemplo se ubica en un nodo del árbol en función de la distancia del valor del atributo utilizado para la partición con respecto a unos valores de referencia (prototipos) calculados previamente. Es decir, las condiciones lógicas sobre el atributo son substituidas por condiciones métricas, de proximidad, sobre dicho atributo. De este modo, cada ejemplo puede estar descrito mediante un conjunto de atributos nominales, numéricos o estructurados, siempre y cuando lleven asociados su correspondiente función de distancia o de similitud. Ilustramos esta propuesta mediante dos ejemplos de clasificación. El primero de ellos referente a un contexto multi-relacional, y el segundo en un contexto con atributos estructurados.

Keyword Tipos de datos estructurados, datos multi-relacionales, árboles de decisión, partición por centros, métodos basados en distancias, métodos basados en instancias.

1. Introducción

En los últimos años, el aprendizaje a partir de datos estructurados ha cobrado una gran importancia. Los nuevos retos del aprendizaje en áreas como la minería Web, la clasificación de textos, la biomedicina, etc. obliga a disponer de lenguajes de representación que ofrezcan soporte para trabajar con tipos de datos estructurados, además de disponer de técnicas que soporten dichas descripciones. En este aspecto, la programación lógico-inductiva (ILP) se ha visto

* Este trabajo ha sido parcialmente subvencionado por el MCyT (SELF TIN 2004-7943-C04-02), la Generalitat Valenciana (GV-04B/477) y la acción integrada hispano-austríaca (h12003-003).

beneficiada de la potencia expresiva proporcionada por el uso de lenguajes de representación basados en lógicas de primer orden, puesto que ofrecen soporte para tipos de datos estructurados elementales como listas o tuplas.

Otra aproximación alternativa para manejar evidencias estructuradas se basa en la utilización de funciones de similitud o de distancia. El concepto de similitud (o de distancia) ha sido explotado en el área del aprendizaje por los métodos basados en instancias, o por los métodos basados en distancias (K-NN o K-means). La idea subyacente en todos ellos es que aquellas evidencias que son similares deberán mantener en común un mayor número de propiedades, y por tanto, deberían de pertenecer a una misma clase o a un mismo grupo según la tarea de aprendizaje que se esté llevando a cabo. Sin embargo, esta familia de técnicas presentan una serie de inconvenientes, ya que en general, no suelen computar un modelo, y en caso de computarlo, éste no es fácilmente comprensible.

Recientemente han surgido diferentes propuestas tanto en el área de la ILP como en el área de la MRDM (minería de datos multi-relacional) que trabajan con datos estructurados o complejos. En [4] se formula un marco de inducción para árboles de decisión basado en lógica de orden superior. El inconveniente principal de esta línea de trabajo es el elevado grado de supervisión por parte del experto. Una variante a esta propuesta, se presenta en [1], donde se aborda el problema mediante técnicas *kernel*, aunque tampoco se computa ningún modelo. En el mismo trabajo se plantea la posibilidad de definir una función de distancia a partir de los *kernels* inferidos y utilizarla, por ejemplo, en un contexto de K-medias. El problema con el que nos encontramos es que dicha función no verifica ciertas propiedades matemáticas exigidas a una métrica y además éstas vienen guiadas por la sintaxis de definición de los tipos, con lo que los patrones a detectar muchas veces no resultan ser muy intuitivos. En [3], se describe una extensión para la inducción de árboles de decisión orientada a abordar problemas de aprendizaje sobre fuentes de datos multi-relaciones, pero los tipos de los atributos soportados son solamente nominales o numéricos (no estructurados), característica que en cierto modo supone una limitación al traducir en un lenguaje de pares atributo-valor las relaciones referenciales existentes en el modelo.

En este trabajo proponemos un marco de inducción que extiende el aprendizaje de árboles de decisión en el sentido que permite abordar de una manera homogénea problemas cuyos atributos sean datos de tipo nominal, numérico, o estructurado. Con este propósito, la única condición es que cada atributo tenga asociado un espacio métrico o una función de similitud. Como se verá en mayor detalle en la sección siguiente, estas funciones se utilizarán para computar un conjunto de representantes (prototipos) por cada uno de los atributos, y efectuar particiones sobre cada uno de estos conjuntos de prototipos, eligiendo como atributo de partición aquél que optimice la función heurística dada. Además, este planteamiento permite aprovechar la definición de métricas y funciones de similitud especializadas sobre diferentes tipos de datos conocidos (listas, árboles, grafos, conjuntos, etc.) ya existentes en la literatura.

Este trabajo se organiza como sigue. En la Sección 2 se describe el método para la inducción de árboles de decisión basados en distancias. La Sección 3 ilustra la aplicabilidad de la propuesta mediante dos sencillos ejemplos. El primero de ellos se centra en el manejo de atributos estructurados, mientras que el segundo aborda un problema de clasificación en un entorno multirelacional. En la Sección 4 se muestran las conclusiones de este trabajo y y se esbozan las líneas de trabajo futuro.

2. Combinando el aprendizaje de árboles de decisión y los métodos basados en distancia

En esta sección presentamos un método de aprendizaje de árboles de decisión que usa prototipos y distancias para definir las particiones del árbol. Esto permite aunar la eficiencia y la comprensibilidad del aprendizaje de árboles de decisión con la capacidad típica de los métodos basados en instancias de trabajar con cualquier tipo de datos estructurado sobre el que se haya definido una función métrica.

2.1. Antecedentes y planteamiento general

La técnica conocida como "partición por centros" (*center splitting* [7]), la cual se ha aplicado a la clasificación de ejemplos representados mediante pares atributo-valor, puede verse como un primer intento de aunar ambos métodos de aprendizaje. Básicamente, en esta técnica se va dividiendo el espacio métrico en diferentes regiones cada una de las cuales se representa mediante un punto especial llamado centro, el cual puede o no pertenecer a la evidencia. En cada iteración del proceso se calcula un centro para cada clase presente en un área. Entonces, cada ejemplo se asigna al centro más próximo. Para cada región impura se repite el proceso hasta que todas las regiones sean puras (es decir, todos los ejemplos en una región sean de la misma clase). Aunque el método es eficiente y flexible, tiene algunos inconvenientes:

- sólo trabaja con atributos nominales y numéricos
- los modelos son poco inteligibles ya que las condiciones resultantes son de la forma "e está próximo al centro $C1$", las cuales para más de dos dimensiones son difícilmente expresables como condiciones sobre los atributos, es decir de la forma $X \leq b$ o $X > b$. Esto se debe a que esta técnica maneja los ejemplos como un todo, es decir, no entra en la estructura del ejemplo para identificar cuál es el atributo que permite resolver el problema de forma adecuada tal y como se hace en el aprendizaje de árboles de decisión.

Con objeto de solventar ambas limitaciones proponemos una nueva estrategia de inferencia de árboles de decisión, que denominamos aprendizaje de árboles de decisión basado en distancias (*Distance-based Decision Tree Learning, DBDT*). DBTD se inspira en el método de las particiones por centro, pero donde las particiones se hacen teniendo en cuenta un atributo cada vez (el "atributo centroide" o prototipo). Por lo tanto, las principales diferencias entre ambas aproximaciones son:

- para computar un prototipo sólo se tienen en cuenta los valores de un atributo mientras que los centros se computan usando todos los atributos cada vez. Para ello es necesario asociar un espacio de similitud a cada atributo (lo que nos permite trabajar con atributos nominales, numéricos y estructurados al mismo tiempo).
- para calcular las particiones con respecto a los prototipos se procede como en el caso de los centros pero considerando únicamente la distribución de las clases y las distancia sobre el atributo seleccionado.

2.2. Algoritmo

El siguiente algoritmo es el procedimiento principal de nuestro método. Toma como entrada un conjunto de entrenamiento S y una constante m que establece el número máximo de hijos por nodo (nótese que si $m = 2$ todas las particiones son binarias) y aprende un árbol de decisión binario. Básicamente es un típico algoritmo de aprendizaje de árboles de decisión pero que computa para cada atributo una lista priorizada de prototipos la cual determina el número de hijos del nodo.

PROCEDURE DBTD(S, m);
// Aprendizaje de un árbol de decisión a partir de las distancias
//definidas en los atributos
INPUT: Un conjunto de entrenamiento S de la forma:
$(x_1, \ldots, x_n), n \geq 1$ donde cada atributo es nominal,
numérico o estructurado. Cada atributo tiene asociado
un espacio métrico. m es el núm. máx. de hijos por nodo.
BEGIN
$C \leftarrow \{Class(e) : e \in S\}$ // C es el conjunto de clases existentes
If $|C| < 2$ **Then RETURN End If**
For cada atributo x_j:
// Computa dos (o más) centros por cada clase utilizando el
atributo x_j.
If $Values(x_j, S) < 2$ **Then CONTINUE End If** //próxima iteración
$ProtList \leftarrow$ Compute_Prototypes(x_j, S, m, C).
If $Size(ProtList) \leq 1$ **Then RETURN End If**
$Split_j \leftarrow \emptyset$ // Conjunto de posibles splits por atributo x_j
For i $\leftarrow$ 1 **to** $length(ProtList)$ // para todos los prototipos
$\hat{S}_i \leftarrow \{e \in S : i =$ Attracts$(e, ProtList, x_j)\}$
// $\hat{S}_i$ contiene los ejemplos atraídos por el prototipo i
$Split_j \leftarrow Split_j \cup \hat{S}_i$ // Añadimos un nuevo hijo a este split
$i \leftarrow i + 1$;
End For
End For
$BestSplit = Argmax_{Split_j}(Optimality(Split_j))$ // GainRatio, MDL, ...
For each set S_k in $BestSplit$

```
    DBTD(S_k, n) // seguir con cada hijo
  End For
END
```

donde

- $Attr_{x_j}(e)$ devuelve el j-ésimo atributo del ejemplo e.
- $class(e)$ devuelve la clase del ejemplo e.
- $Values(S, x_j)$ devuelve el número de valores diferentes del atributo x_j en la evidencia S.
- $Card(v, S, x_j)$ devuelve el número de ocurrencias del valor v en el atributo x_j en la evidencia S, esto es, $|\{e \in S : Attr_{x_j}(e) = v\}|$.
- $dist(x, y)$ computa la distancia de los valores x e y, siendo x e y atributos del mismo tipo.

Las dos funciones más significativas del algoritmos `DBTD` son `Compute_Prototypes` y `Attracts`. La función `Attracts` asigna a un nuevo ejemplo el prototipo más próximo, y en caso de que haya empate el que está situado más a la derecha en la lista priorizada.

FUNCTION `Attracts(`$e, ProtList, x_j$`);`
`// Devuelve el prototipo más cercano al ejemplo.`
`// En caso de empato, devuelve el prototipo de más a la derecha.`
INPUT: `Un ejemplo` e. `Una lista priorizada de prototipos` $Protlist$.
`Un atributo elejido,` x_j
RETURNS: `Los índices de los prototipos que atraen al ejemplo` e.
BEGIN
  For $i \leftarrow 1$ **to** $length(ProtList)$
    $v \leftarrow attr_{x_j}(e)$
    If $\forall k > i, dist(attr_{x_j}(ProtList[i]), v) < dist(attr_{x_j}(ProtList[k]), v)$ **Then**
      RETURN i
    End If
    $i \leftarrow i + 1$
  End For
END

La función `Compute_Prototypes` selecciona el mejor prototipo, es decir, aquél que minimiza las distancias de los ejemplos de su misma clase a él. Una vez seleccionado, se eliminan tanto la clase como los valores del atributo y continúa buscando el siguiente mejor prototipo para una clase y valor diferentes del atributo. Este proceso se repite hasta alcanzar el límite m.

FUNCTION `Compute_Prototypes(`x_j, S, m, C`);`
`// Computa hasta` m `prototipos para el atributo` x_j `sobre el conjunto de datos` S.
INPUT: x_j `es el atributo.` S `es el conjunto de datos.` m `es el máximo`
`#`

```
        de prototipos. C es el conjunto de clases.
RETURNS: Una lista priorizada de prototipos..
BEGIN
   For cada clase c ∈ C:
     S_c ← {e ∈ S : class(e) = c}. // Ejemplos de clase c
     If S_c ≠ ∅ Then
       V_c ← Values(x_j, S_c)
       For cada elemento v ∈ V_c
         MeanDistance_c[v] ← (Σ_{i∈V_c} dist(v,i)×Card(i,S_c,x_j)) / |S_c|
       End For
     End If
   End For
   UV ← ∅ // Valores del atributo que se ha utilizado
   ProtList ← ∅ // Lista de prototipos
   RC ← C // Clases restantes
   Values ← Values(x_j, S) // Valores diferentes en S
   #Prots ← min(|C|, m, Values) // #Prototipos de la partición
   For k ← 1 to #Prots
     // Selecciona el mejor prototipo cuyo valor de atributo
     // no ha sido ya elegido y para una clase nueva.
     BestProt ← argmin_e{MeanDistance_c[Attr_{x_j}(e)]}_{c∈RC, e∈S, Attr_{x_j}(e)∉UV}
     // En caso de empate, seleccionar aquella con más ejemplos.
     // En caso de nuevo empate, la primera en el orden dado.
     RC ← RC − Class(BestProt)
     ProtList ← append(ProtList, BestProt) // añadir al final
     UV ← UV ∪ Attr_{x_j}(BestProt)
     k ← k + 1
   End For
   RETURN ProtList
END
```

3. Ejemplos

En esta sección presentamos dos ejemplos sobre datos estructurados que ilustran la aplicación del algoritmo descrito anteriormente, el cual se halla implementado utilizando como soporte las librerías de libre distribución del Weka [8].

3.1. Clasificación sobre datos estructurados

El problema de clasificación molecular que proponemos a continuación se puede resumir como sigue. Disponemos de un conjunto formado por tres tipos de moléculas diferentes. Cada molécula consiste de una cadena con un número variable de átomos. Los átomos pueden ser de dos tipos, que denotaremos por

las letras minúsculas b y c, siendo los del tipo c los más comunes. A su vez, cada cadena está encabezada por un grupo molecular. En el conjunto de muestras contamos con cuatro grupos moleculares diferentes, donde cada uno de estos grupos se representa mediante las letras mayúsculas X, Y, Z y T. Además, de cada molécula se conoce también la temperatura de ruptura (medida en $C°$), esto es, la temperatura a partir de la cual el grupo molecular se separa de la cadena de átomos. En la tabla 3.1 se puede apreciar el conjunto de evidencias, las cuales vienen descritas por dos atributos, la temperatura (numérico) y la molécula (estructurado).

id	Temp.($C°$)	Molécula	Clase	id	Temp.($C°$)	Molécula	Clase
1	30	Xc^5	1	10	46	Zc^7	2
2	31	$Xc^{35}b$	1	11	47	Zc^9	2
3	29	$Xc^{35}bc^5bc^5$	1	12	49	Tc^{14}	2
4	33	$Yc^{27}bc^{13}bc^{10}$	1	13	55	$Tc^{30}bc^{10}bc^{10}$	3
5	34	$Yc^{33}bc^7$	1	14	49	Zc^{25}	3
6	31	Yc^5	1	15	50	$Zc^{29}b$	3
7	51	Zc^5	2	16	48	$Tc^{30}bc^7$	3
8	53	Tc^{12}	2	17	52	$Tc^{31}b$	3
9	46	Zc^{10}	2	18	53	$Tc^{28}bc^{12}$	3

Cuadro 1. Conjuntos de ejemplos para el problema de la clasificación molecular.

El próximo paso consiste en formular los espacios métricos asociados a cada uno de los atributos, de manera que las funciones de distancia que de ellos se deriven se ajusten a las necesidades del problema. Aunque para este ejemplo artificial que estamos tratando, las métricas a emplear parecen bastante claras (el valor absoluto de la diferencia para el atributo numérico y la distancia de alineamiento para las moléculas [6], puesto que el dominio del atributo "molécula" se corresponde simplemente con una lista finita de símbolos pertenecientes al alfabeto $\{b, c, T, X, Y, Z\}$), por lo general, esta fase no resultará trivial, y en muchos contextos se puede exigir la definición de una métrica capaz de "discriminar" en base a una serie de reglas o patrones específicos.

Sin perder de vista que el objetivo de este proceso es obtener un árbol de decisión basado en distancias, necesitamos definir una función heurística a partir de la cual obtener un criterio para seleccionar el atributo que efectúa la partición óptima. Por simplificar nuestra traza, escogeremos la precisión de la partición [1] como criterio de selección. En este ejempo estableceremos un número máximo de centroides por atributo que coincida por el número de clases diferentes que haya en un nodo. Como inicialmente todas las evidencias se hallan agrupadas

[1] Dado un nodo e del árbol de decisión, la precisión de una partición se expresa como $\sum_{i=1}^{n} \frac{C_{n_i}}{N}$, donde N es la cardinalidad de e (el número de evidencias en e), n es el número de hijos de e, y C_{n_i} es la cardinalidad de la clase mayoritaria del hijo i-ésimo.

en el nodo raíz, el número de clases diferentes es tres. Agrupando las evidencias por cada clase tenemos,

$$\begin{aligned}\text{Clase 1 (Temp.)} &= \{30, 31, 29, 33, 34, 31\}\\ \text{Clase 2 (Temp.)} &= \{51, 53, 46, 46, 47, 49\}\\ \text{Clase 3 (Temp.)} &= \{55, 49, 50, 48, 42, 53\}\end{aligned}$$

La temperatura que minimiza la suma de las distancias para la clase 1 es $31C^\circ$ (correspondiente al ejemplo 2), $51C^\circ$ para la clase 2 (correspondiente al ejemplo 7), y finalmente $50C^\circ$ para la clase 3 (correspondiente al ejemplo 15). El proceso se repite para la clase "molécula". Después de efectuar los cálculos oportunos, obtenemos el ejemplo 2 como centroide de la clase 1, el ejemplo 9 como centroide de la clase 2, y finalmente, el ejemplo 16 como centroide de la clase 3. Ahora seleccionamos la partición más precisa. En este caso, la mejor partición viene dada por el atributo "temperatura" (una precisión de 0,723 frente a una del 0,667 para el atributo molécula). Esta partición, tal como se puede apreciar en la figura 1, origina tres nodos hijos siendo sólo el correspondiente a la condición "temperatua próxima a $31C^\circ$", puro. Las condiciones métricas sobre los atributos numéricos se traducen por condiciones lógicas de la siguiente manera. Se obtienen los extremos (valores más alejados del centroide) de cada uno de los grupos, y se calcula su semisuma. Así pues, para el nodo 1 cuyo centroide es la evidencia número 2 con una temperatura de $31C^\circ$, y el nodo 3 que tiene como centroide la evidencia número 15 con una temperatura de $50C^\circ$, tienen como valores extremos evidencias con temperaturas de $34C^\circ$ y $42C^\circ$ respectivamente. Calculando su semisuma, obtenemos un valor de $40C^\circ$. Por lo tanto, la condición métrica anterior se reescribe a "temperatura $\leq 40C^\circ$".

Ahora, tenemos que volver a iterar el proceso sobre los nodos no puros. El mejor atributo para efectuar la partición es el atributo "molécula", siendo los centroides computados las evidencias 8 y 17 para el nodo 2 y 11 y 15 para el nodo 3. Al aplicar las particiones sobre los nodos 2 y 3 por el atributo "molécula" obtenemos nuevos nodos que son puros. La Figura 1 ilustra el árbol de decisión inducido.

Como se puede observar en este ejemplo, las particiones sobre atributos estructurados son más complejas que las particiones efectuadas sobre atributos nominales o numéricos. Sin embargo, gracias a que el proceso ha estado guiado por un atributo clave (el atributo seleccionado por la heurística), resulta más fácil la representación del modelo en base a reglas, que si durante el proceso de inferencia los ejemplos se hubieran considerado como un todo. Del árbol de decisión anterior se podría derivar el siguiente conjunto de reglas:

- La clase 1 se corresponde con aquellas moléculas cuya temperatura de ruptura es menor o igual que $40C^\circ$.
- La clase 2 se corresponde con aquellas moléculas que poseen una cadena de átomos "corta" y cuya temperatura es superior o igual a $50{,}5C^\circ$
- La clase 3 se corresponde con aquellas moléculas que poseen una cadena de átomos "larga" y cuya temperatura oscila entre $40C^\circ$ y $50{,}5C^\circ$.

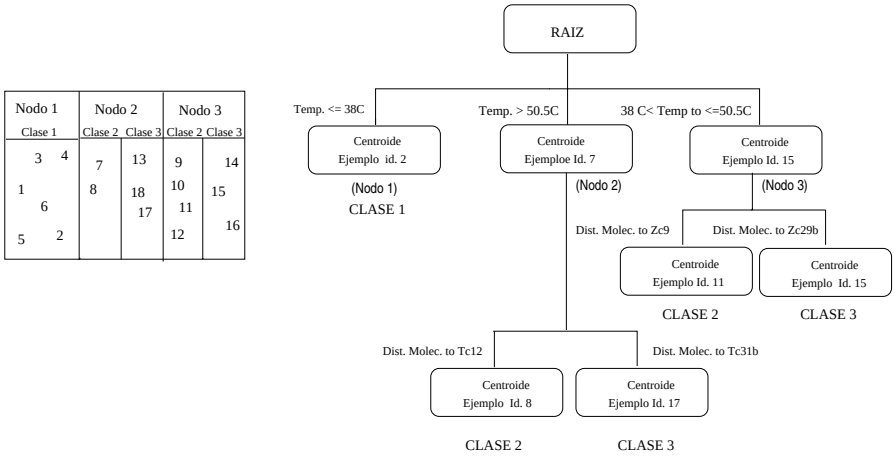

Figura 1. La figura de la izquierda representa la división del espacio de evidencias. La figura de la derecha se corresponde con el árbol de decisión inducido.

Nótese que la métrica empleada en los tipos estructurados resulta primordial para la detección de patrones que pueden resultar claves para la clasificación. La sustitución de las condiciones métricas por patrones que faciliten la interpretación del modelo obtenido constituye una de las líneas de trabajo futuro, tal y como se comenta en las conclusiones.

3.2. Clasificación sobre datos multi-relacionales

En este ejemplo explicamos cómo aplicar nuestro método para la extracción de un modelo desde una base de datos multi-relacional. El problema consiste en clasificar a los empleados de una organización como buenos o malos trabajadores en función de la información presente en el repositorio. Dicho repositorio consta de cuatro tablas (*Employee, Participates, Project y Department*) organizadas tal y como se muestran en la figura 2, reflejando la información personal de cada trabajador además de los proyectos en los que participa y del departamento al que pertenece. Como información personal de cada trabajador merece destacar el atributo *profile* que muestra la formación y trayectoria profesional (estudios uninversitarios (D), masters (M), doctorado (PhD), programador (Pr), etc.). El modelo relacional no soporta la inclusión directa de tipos estructurados, por lo que este atributo se modela como una cadena de caracteres, aunque el sistema del aprendizaje lo usará como un atributo estructurado (lista).
A continuación, supóngase que un usuario selecciona la clave ajena desde *Employee* hacia *Department* y las claves ajenas desde *Participates* hacia *Employee* y *Project* para demarcar el ámbito de aprendizaje. Este hecho involucra a las cuatro tablas en el esquema. Es necesario establecer qué información (atributos y tablas) serán fundamentales para abordar el problema de la clasificación de

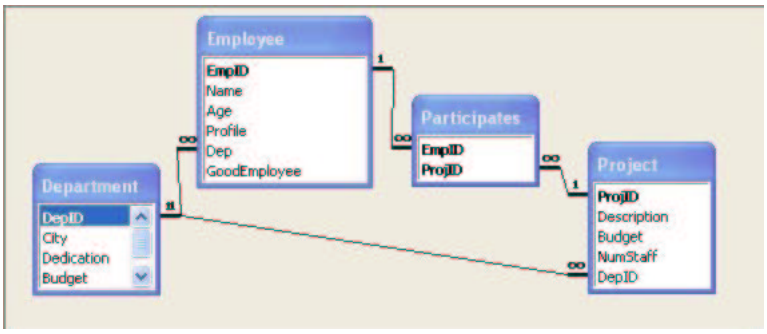

Figura 2. Base de datos relacional con múltiples tablas y claves ajenas.

los empleados en buenos o malos. Por ejemplo, en la tabla *Department* sólo los atributos *Budget* y *Outsourcing* parecen ser relevantes.

Si deseamos resolver este problema mediante la utilización de algoritmos clásicos de árboles de decisión, debemos describir cada ejemplo en una representación atributo-valor. Esto implica la desnormalización del esquema relacional, y en esta situación, surgen dos problemas habituales: Algunos datos serán redundantes (por ejemplo, un empleado aparecerá en la tabla tantas veces como el número de proyectos en el que esté incluido) y el número de atributos por tupla será innecesariamente alto (la información relativa a los proyectos y departamentos). Ambos problemas afectarán de manera negativa a la calidad y la eficiencia del proceso de aprendizaje.

Nuestra aproximación utiliza directamente la base de datos, sin necesidad de desnormalizar el esquema. Con este propósito, empleamos atributos con datos estructurados. Concretamente, todas las relaciones múltiples se representan mediante un atributo del tipo conjunto. Por ejemplo, la clave ajena desde *Participates* hacia *Employee* se representa como un atributo adicional de tipo conjunto insertado en la tabla *Employee*, denominado *Projects*, y cuyos elementos contienen referencias a las tuplas de la tabla *Project*. Por lo tanto, además de los atributos estructurados existentes (por ejemplo *Profile* es una lista), otros atributos estructurados nuevos pueden aparecer al analizar el esquema relacional.

En nuestra aproximación, todos los atributos de tipo estructurado son tratados de idéntica manera. Cada valor de estos atributos se codifica mediante un identificador. Este identificador representa el número de fila en una matriz que contiene las distancias entre cada diferente valor existente en los datos de entrenamiento de ese determinado tipo (matriz de distancias). De esta manera, la distancia entre dos objetos se calcula sólo una vez, y siempre que se necesite utlizar esa distancia, se puede recuperar de la matriz de distancias. En este contexto, una clave ajena puede ser también considerada como el identificador de un atributo estructurado, ya que se refiere a su valor: una tupla.

Con la transformación previa, precalculando las distancias necesarias, nuestro método `DBDT` retorna el siguiente árbol de decisión:

```
Projects CLOSE TO {Fraud Solutions,Increasing Sales}
|  Profile CLOSE TO [D,M,De,An]
|     30<=Age<=49 : True
|     50<=Age<=60
|           Profile CLOSE TO [D,Ph,Re] : False
|           Profile CLOSE TO [D,M,De,An] : True
|  Profile CLOSE TO [D,Re] : False
Projects CLOSE TO {Fraud Solution, Increasing Sales,
                           Work Performance} : False
```

El conjunto previo de reglas muestra cómo algunas particiones contienen condiciones acerca de la similitud entre valores de un atributo (por ejemplo los atributos correspondientes a listas o conjuntos). Las particiones que corresponden a valores numéricos contienen los tests típicos.

Finalmente, concluimos esta sección comparando el modelo previo con el modelo aprendido con el algoritmo ID3. Dado que este algoritmo requiere que los datos estén contenidos en sólo una tabla, hemos desnormalizado el esquema relacional de manera estándar (incluyendo departamentos y proyectos, y repitiendo las tuplas de los empleados cuando se requiera). Adicionalmente, hemos discretizado el atributo *Age* y hemos utilizado el atributo *Profile* como nominal. ID3 genera el siguiente árbol de decision:

```
Profile = 0
  Department = 0: null
  Department = 1
    Project = 1
      Age = 30-40: True
      Age = 40-50: True
      Age = 50-60: True
    Project = 2
      Age = 30-40: True
      Age = 40-50: True
      Age = 50-60: False
      Project = 3
      Age = 30-40: True
      Age = 40-50: False
      Age = 50-60: True
    Project = 4: null
  Department = 2: False
Profile = 2: False
```

```
Profile = 1
  Age = 30-40: True
  Age = 40-50: True
  Age = 50-60: True
Profile = 3: False
Profile = 4:
  Project 1
   Age = 30-49: False
   Age = 40-50: True
   Age = 50-60: null
  Project = 3: False
    Project = 4: True
```

```
Profile = 5
 Project = 1
   Age = 30-40
    Department = 0 : null
    Department = 1: False
    Department = 2: True
   Age = 40-50: True
   Age = 50-60: False
Project = 2 : False
Project = 3
   Age = 30-40
    Department = 0 : null
    Department =  1: False
    Department =  2: True
   Age = 40-50: True
   Age = 50-60: False
Project = 4: null
```

Puede ocurrir, sin embargo, que existan algunos casos, para los que una desnormalización completa sea una mejor opción. Pero, en este ejemplo, lo que deseamos mostrar es la gran expresividad del árbol generado mediante nuestra aproximación con respecto al árbol generado con métodos proposicionales clásicos como ID3.

4. Conclusiones y trabajo futuro

En este trabajo, hemos presentado un algoritmo que combina el aprendizaje de árboles de decisión y el aprendizaje basado en distancias. La clave del método propuesto es que utiliza centroides basados en un único atributo para generar las particiones del árbol de decisión. De este modo somos capaces de inducir modelos, específicamente árboles de decisión, que utilizan la distancia asociada al tipo de datos del atributo seleccionado por el criterio de partición. Consecuentemente, el algoritmo puede abordar problemas con instancias que

contengan atributos de cualquier tipo de datos. El único requisito es que el tipo debe tener asociado una función de distancia o métrica. Por otra parte, dado que calculamos las distancias entre atributos, en vez de entre ejemplos completos, la eficiencia del proceso de aprendizaje del árbol de decisión se preserva parcialmente. Mejor aún, podemos utilizar matrices de distancias para almacenar las distancias más utilizadas incrementando la eficiencia del aprendizaje.

La ventaja principal de nuestra aproximación con respecto a otras técnicas de aprendizaje clásicas de árboles de decisión, es la capacidad de tratar con problemas que contengan atributos con tipos complejos tales como listas o conjuntos. Con respecto a otras aproximaciones al aprendizaje multi-relacional tales como RIBL [2], nuestro algoritmo presenta la ventaja que realmente infiere un modelo que puede ser expresado como un árbol de decisión, y por lo tanto, un usuario puede comprender, o al menos tener una idea, de cómo el modelo clasifica nuevos ejemplos.

Como trabajo futuro de carácter ambicioso nos planteamos estudiar cómo convertir las particiones basadas en distancias de los atributos de tipo estructurado en particiones basadas en patrones, de manera que sean fácilmente comprensibles. Aunque este proceso no parece difícil para atributos de tipo nominal o numérico, y el resultado sería similar a un típico árbol de decisión, no ocurre lo mismo para los tipos de datos más complejos. Con este propósito, podríamos definir un conjunto de propiedades asociadas a cada tipo de datos, por ejemplo, para listas podríamos utilizar el siguiente conjunto de propiedades: cabeza, cola, miembro, etc. Otra aproximación podría ser las adaptaciones del operador de generalización menos general (lgg) de Plotkin [5].

Referencias

1. T. Gärtner, J. W. Lloyd, and P. A. Flach. Kernels for structured data. In *Proceedings of the 12th International Conference on Inductive Logic Programming*, volume 2583 of *LNAI*, pages 66–83, 2003.
2. Tamas Horvath, Stefan Wrobel, and Uta Bohnebeck. Relational instance-based learning with lists and terms. *Machine Learning*, 43(1/2):53–80, April 2001.
3. A. J. Knobbe, A. Siebes, and D. M. G. Van der Wallen. Multi-relational decision tree induction. In *Proceedings of the 3d European Conference on Principles of Data Mining and Knowledge Discovery*, pages 378–383. Springer-Verlag, September 1999.
4. J. W. Lloyd. *Logic for learning: learning comprehensible theories from structured data.* Springer-Verlag, 2003.
5. G. Plotkin. A further note on inductive generalization. *Machine Intelligence*, 6, 1971.
6. T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
7. Chris Thornton. *Truth from Trash: How Learning Makes Sense.* The MIT Press, Cambridge, Massachusetts, 2000.
8. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann Publishers, 1999.