Inducción de árboles de decisión con CIDIM: nuevos enfoques *

Gonzalo Ramos-Jiménez, José del Campo-Ávila y Rafael Morales-Bueno

Departamento de Lenguajes y Ciencias de la Computación

E.T.S. Ingeniería Informática. Universidad de Málaga

Málaga, 29071, España

{ramos, jcampo, morales}@lcc.uma.es

Resumen

En este artículo presentamos CIDIM (Control de Inducción por DIvisión Muestral), un algoritmo que ha sido desarrollado para inducir árboles de decisión precisos y pequeños. Para conseguirlo, intenta reducir el sobreajuste usando un control de inducción local. Otras ideas como la división dicotómica del conjunto de experiencias o la agrupación de valores consecutivos ayudan a mejorar el comportamiento del algoritmo. CIDIM ha sido comparado de forma satisfactoria con C4.5 puesto que induce árboles de decisión que suelen ser más precisos y, sobre todo, más pequeños. En este artículo, presentamos además, dos variantes de sistemas multiclasificadores que usan a CIDIM como algoritmo básico de aprendizaje: M-CIDIM y E-CIDIM.

1. Introducción

El aprendizaje a partir de experiencias y la inducción de conocimiento a partir de ellas conforman un área de investigación muy extensa en el campo del aprendizaje automático (Machine Learning). Se han propuesto muchos modelos para representar dicho conocimiento, pero uno de los más usados es el árbol de decisión (CART [2], ID3 [9], C4.5 [10], ITI [14], VFDT [3] ...). De entre todas las características positivas que tiene, nos gustaría destacar su capacidad de dividir el espacio de experiencias en subespacios y ajustar cada uno de ellos con distintos modelos, además de la comprensibilidad del modelo generado.

Los árboles de decisión suelen evaluarse atendiendo a la precisión que se consigue con ellos y a su tamaño [4]. Es deseable conseguir árboles de decisión que sean precisos, pero, si además conseguimos que sean pequeños, mejor. Es bien sabido que el problema de encontrar el árbol más pequeño que clasifique un conjunto de experiencias es un problema $NP - completo$ [7, 5] y, por lo tanto, diseñar una buena estrategia de expansión para el árbol se convierte en algo fundamental.

En este artículo presentamos el algoritmo CIDIM [12, 11] (Control de Inducción por DIvisión Muestral), un algoritmo que induce árboles de decisión usando un control local para la inducción. Éste control, combinado con una división dicotómica del conjunto de experiencias de entrenamiento y la agrupación de valores consecutivos, permite a CIDIM inducir árboles precisos y pequeños.

Además de CIDIM, también describimos dos algoritmos multiclasificadores que aprovechan las cualidades de CIDIM: M-CIDIM, que mantiene un esquema similar al bagging, pero modifica la forma en la que se perturba el conjunto de experiencias y E-CIDIM en el que existe un proceso de generación de árboles que van sustituyendo a los previos siempre que sean mejores.

El artículo se organiza de la siguiente forma: en la Sección 2 se describe el algoritmo CIDIM, comentando sus principales características. M-CIDIM y E-CIDIM se explican brevemente en

*Este trabajo ha sido financiado parcialmente por el proyecto MOISES, número TIC2002-04019-C03-02, del MCyT, España.

Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA2005, pp.13-20
ISBN: 84-9732-449-8

las Sección 3. Los resultados de algunos experimentos se comentan en la Sección 4 y para terminar presentaremos las conclusiones y algunos trabajos futuros en la Sección 5.

2. CIDIM

CIDIM [12, 11] (Control de Inducción por DIvisión Muestral) ha sido desarrollado con la intención de construir árboles de decisión precisos y pequeños. Puede usarse para extraer conocimiento de problemas con un número finito de atributos. Estos atributos deben ser nominales y pueden estar ordenados o no. Para tratar atributos continuos tenemos dos opciones: discretizarlos en intervalos o tomar cada uno de los distintos valores que aparecen en el conjunto de experiencias como uno de los valores del atributo.

Existen tres ideas básicas que fundamentan el comportamiento de CIDIM y que explicaremos de forma detallada en las siguientes subsecciones: la división del conjunto de entrenamiento, la agrupación de valores consecutivos y un control local para detener la inducción.

2.1. División del conjunto de entrenamiento

Los algoritmos clásicos para la inducción de árboles de decisión (TDIDT [9, 10]) suelen dividir el conjunto de experiencias en dos subconjuntos: el subconjunto de entrenamiento (con el que se induce el árbol) y el subconjunto de test (con el que se comprueban los resultados). CIDIM realiza una división adicional sobre el subconjunto de entrenamiento (E) y obtiene dos nuevos subconjuntos con la misma frecuencia de clases y de tamaño similar. De los nuevos conjuntos usamos uno para la construcción (llamado CNS) y otro para las tareas de control (llamado CLS). Se puede observar una descripción más formal de estos subconjuntos en la Figura 1.

CNS y CLS se usan en el proceso de expansión. Cada nodo tiene sus correspondientes subconjuntos CNS y CLS y, cuando se realiza una expansión, éstos subconjuntos se dividen de forma adecuada atendiendo a los nuevos hijos que se acaben de generar. De esta forma, el

$$
\begin{array}{l}
|CNS| \neq 0 \quad \wedge \quad |CLS| \neq 0 \\
CNS \cap CLS = \emptyset \\
CNS \cup CLS = E \\
0 \leq \Big|\{e \in CNS \mid C(e) = i\}\Big| - \\
\qquad \Big|\{e \in CLS \mid C(e) = i\}\Big| \leq 1 \\
\qquad\qquad\qquad \forall i \in CLASES
\end{array}
$$

donde $C(e)$ es la clase de la experiencia e

Figura 1: Subconjuntos CNS y CLS

tamaño de ambos conjuntos va disminuyendo conforme profundizamos en el árbol. Lo importante de estos subconjuntos radica en el uso que se hace de ellos para formar grupos de valores consecutivos (ver Subsección 2.2) y para evaluar la condición de control local a cada nodo en el proceso de expansión (ver Subsección 2.3).

2.2. Agrupación de valores consecutivos

Normalmente, cuando se expande un nodo, se añade un hijo por cada uno de los valores de un atributo nominal. Pero si el atributo es ordenado, sería deseable hacer un tratamiento previo diferente.

CIDIM contempla distintos comportamientos para realizar la expansión en función del tipo de atributo por el que vaya a expandir. Si el atributo no es ordenado, la expansión se mantiene como en la forma clásica (un hijo por cada valor). Pero si el atributo es ordenado, CIDIM usa un algoritmo voraz para encontrar agrupaciones de valores para dicho atributo. Dicho algoritmo está basado en una partición dicotómica recursiva de los valores en grupos. Inicialmente existe un único grupo con todos los valores y en cada paso se evalua si una división en dos grupos produciría alguna mejora. El proceso continúa hasta que no haya mejora o hasta que sólo quede un valor por grupo. Para realizar las divisiones se usa el subconjunto CNS y se decide si una división produce alguna mejora mediante una medida de desorden (que llamaremos $md\_division$). Usando esta agrupación, CIDIM busca cuál es la mejor división para cada uno de los atributos que que-

dan sin usar. Al igual que hace el algoritmo BOAT [6], CIDIM no se limita a usar siempre la misma medida de desorden y ésta puede configurarse.

Otra medida de desorden (md_mejora) se usa para decidir qué par atributo-división es el mejor de todos los posibles en el nodo que se va a expandir. Esta medida de desorden también puede configurarse y puede ser la misma que la usada para la división. Tras una serie de pruebas empíricas hemos determinado que la entropía es una buena medida de desorden para ser usada en el proceso de división. Sin embargo, para el proceso de mejora no hemos identificado ninguna que dé los mejores resultados.

Cuando se ha elegido el mejor par atributo-división, se realiza un control local de inducción (ver Subsección 2.3) para evaluar la conveniencia de realizar la expansión correspondiente. Este control es local a cada nodo puesto que un mismo atributo puede tener distintas agrupaciones de valores en función del nodo en el que se encuentre.

2.3. Control local de inducción

La forma más sencilla para detener la expansión de los árboles es hacerlo cuando todas las experiencias que están en un nodo hoja se etiquetan con la misma clase, pero esto suele generar árboles demasiado grandes. Para evitar esto, algunos algoritmos introducen un criterio externo que detiene la ejecución (por ejemplo, C5, una versión mejorada de C4.5, necesita que al menos dos ramas tengan como mínimo, un número preconfigurado de experiencias). CIDIM usa la siguiente condición local: un nodo se expande sólo si la expansión mejora la precisión calculada sobre el subconjunto de control (CLS). Esta condición se supervisa de forma local en cada nodo y tiene en cuenta dos índices: un índice absoluto (I_A) y otro relativo (I_R) (ver ecuaciones (1) y (2)). Un nodo sólo se expande si alguno de los dos índices mejora y ninguno empeora. Los índices absoluto y relativo se definen del siguiente modo:

$$I_A = \frac{\sum_{i=1}^{N} CORRECT(e_i)}{N} \qquad (1)$$

$$I_R = \frac{\sum_{i=1}^{N} P_{C(e_i)}(e_i)}{N} \qquad (2)$$

donde N es el número de experiencias en CLS, e es una experiencia, $C(e)$ es la clase de la experiencia e, $P_m(e)$ es la probabilidad de la clase m para la experiencia e en el nodo usando el subconjunto CNS, y $CORRECT(e) = 1$ si $P_{C(e_i)} = max\{P_1(e), P_2(e), ..., P_k(e)\}$ o 0 en otro caso.

2.4. Algoritmo

Una vez que hemos presentado los componentes fundamentales del algoritmo podemos describirlo con más detalle. A continuación mostramos el pseudocódigo del algoritmo.

Entrada:
E, $md\_division$, md_mejora

$\{CNS, CLS\} = Div\_Dicot\_Aleat(E)$
$Raiz = Nuevo\_Nodo(CNS, CLS)$
$SinEtiq = \{Raiz\}$
mientras $SinEtiq \neq \emptyset$ **hacer**
  $Nodo = Selec\_Aleat(SinEtiq)$
  $Atrib\&Div = Mejor\_Atrib\_Div$
    $(Nodo, md\_division,$
    $md\_mejora)$
  si $Mejora\_Pred(Nodo,$
    $Atrib\&Div)$ **entonces**
    $Hijos = Expandir(Nodo, Atrib\&Div)$
    $SinEtiq = SinEtiq - \{Nodo\}$
    $SinEtiq = SinEtiq \cup Hijos$
  si no
    $SinEtiq = SinEtiq - \{Nodo\}$

Salida:
$Raiz$

Algoritmo 1: Algoritmo CIDIM

El algoritmo CIDIM induce un árbol de decisión cuyo nodo raíz es devuelto ($Raiz$) y para hacerlo necesita un conjunto de experiencias de entrenamiento (E) y dos medidas de desorden ($md\_division$ y md_mejora).

El primer paso del algoritmo es una división dicotómica aleatoria del conjunto de entrenamiento ($Div\_Dicot\_Aleat(E)$) en dos sub-

conjuntos (CNS y CLS). Esta división mantiene la frecuencia de clases del conjunto original (E) y cumple los requisitos expuestos en la Figura 1. Después de realizar la división, se construye la raíz del árbol usando los subconjuntos CNS y CLS. Desde ese momento se repite un proceso iterativo que no termina mientras queden nodos sin etiquetar ($SinEtiq \neq \emptyset$). El proceso comienza seleccionando un nodo sin etiquetar de forma aleatoria ($Nodo = Selec\_Aleat(SinEtiq)$) y calculando el par atributo-división que mayor mejora produzca ($Atrib\&Div = Mejor\_Atrib\_Div(\ Nodo, md\_division, md\_mejora)$). Dicho par se calcula usando el subconjunto de construcción (CNS) y las dos medidas de desorden según se explica en la Subsección 2.2. Cuando se determina dicho par, CIDIM comprueba si la mejor expansión que se puede realizar mejora la predicción dada por el nodo sin expandir. Si la expansión no se debe realizar, el nodo se etiqueta como hoja; pero si la expansión mejora la predicción, se añaden al árbol los hijos correspondientes al atributo seleccionado según la división realizada.

Debemos comentar que una versión previa y simplificada de CIDIM [12] ha sido utilizada para resolver problemas reales como modelado de sistemas [13] o modelado para el pronóstico de la reaparición de cáncer de mama [8].

3. Multiclasificadores basados en CIDIM

Una de los objetivos del aprendizaje automático es mejorar la generalización de los clasificadores, es decir, conseguir que los modelos se ajusten mejor al conocimiento de forma que aumente la precisión alcanzada con ellos. Los sistemas multiclasificadores y los métodos de votación consiguen avanzar en estos aspectos. Hemos usado las particularidades de CIDIM para elaborar dos sistemas multiclasificadores que describiremos brevemente: M-CIDIM y E-CIDIM.

3.1. M-CIDIM

M-CIDIM está basado en el esquema de bagging pero incluye la particularidad de que la perturbación que se realiza en el proceso está forzada con la intención de aprovechar las características propias de CIDIM. Este nuevo esquema lo hemos llamado Forbagging (Bagging Forzado) y se presenta en el Algoritmo 2.

La principal característica de este esquema es que la perturbación de cada paso se hace teniendo en cuenta la perturbación que se hizo en el paso anterior ($E(i) = Perturba(E(i-1))$) y así podemos incluir una tendencia que intente mejorar el comportamiento de bagging. Si contamos con un algoritmo que divide el conjunto de entrenamiento en dos, uno para construcción y otro para control (como es el caso de CIDIM), puede que nos interese guiar de alguna forma la división. M-CIDIM usa este esquema para hacer que las sucesivas divisiones sean lo más diferente posible respecto de la anterior y de esta forma intentamos aprender diferentes zonas del espacio de experiencias. Las divisiones realizadas por M-CIDIM se hacen de forma ortogonal: el conjunto de construcción se divide en dos subconjuntos (manteniendo la frecuencia de clases) y el de control también; los conjuntos de construcción y control del siguiente paso se construyen usando una mitad de cada uno de los conjuntos anteriores.

Entrada:
conjunto de entrenamiento E,
clasificador básico C,
tamaño del sistema multiclasificador n

$E(0) = E$
desde $i = 1$ **hasta** n **hacer**
$E(i) = Perturba(E(i-1))$
$M(i) = C(E(i))$
$M = Combina(M(1), ..., M(n))$

Salida:
Sistema multiclasificador M

Algoritmo 2: Algoritmo M-CIDIM

3.2. E-CIDIM

E-CIDIM también aprovecha las características propias del CIDIM para construir el multiclasificador. E-CIDIM tiene dos parámetros para configurarlo: el número máximo de árboles en el multiclasificador (Max_arb) y el número de intentos fallidos antes de detener la inducción ($Max\_intentos$). Además de estos parámetros, se debe seleccionar el tipo de votación que se va a usar.

Entrada:
E, Max_arb, $Max\_intentos$

$MultiCl = \emptyset$
$Arb\_iniciales = \lceil Max\_arb/2 \rceil$
desde 1 **hasta** $Arb\_iniciales$ **hacer**
  $Nuevo\_arb \leftarrow$ Induce nuevo árbol con CIDIM usando E
  $MultiCl = MultiCl \cup Nuevo\_arb$
$NoMejora = 0$
while $NoMejora < Max\_intentos$ **hacer**
  $Peor\_arb = \{x \in MultiCl \mid ind\_acierto(x) < ind\_acierto(y) \; \forall y \in MultiCl \wedge x \neq y\}$
  $Nuevo\_arb \leftarrow$ Induce nuevo árbol con CIDIM usando E
  si $ind\_acierto(Nuevo\_arb) > ind\_acierto(Peor\_arb)$ **entonces**
    $NoMejora = 0$
    $MultiCl = MultiCl \cup Nuevo\_arb$
    si $|MultiCl| > Max\_arb$ **entonces**
      $MultiCl = MultiCl - Peor\_arb$
  si no
    $NoMejora = NoMejora + 1$

Salida:
$MultiCl$

Algoritmo 3: Algoritmo E-CIDIM

En primer lugar, E-CIDIM inicializa el conjunto ($MultiCl$) con algunos árboles de decisión inducidos por CIDIM usando el conjunto de entrenamiento (E). Normalmente, los sucesivos árboles de decisión que se van generando serán diferentes unos de otros, puesto que la partición dicotómica del conjunto de entrenamiento es aleatoria y se realiza en cada ejecución de CIDIM. Tras la inicialización tenemos la mitad de los árboles de decisión que se indicaron como parámetro del máximo posible (Max_arb). A partir de entonces induciremos nuevos árboles que podrán irse añadiendo al conjunto. Si el nuevo árbol inducido en cada paso ($Nuevo\_arb$) tiene un índice de acierto mejor que el del peor árbol en el conjunto ($Peor\_arb$), se añade al conjunto. Si ya hubiésemos llegado a tener en el conjunto el máximo número de árboles permitido y hubiese que añadir algún árbol más, el peor nodo sería eliminado del conjunto. E-CIDIM intenta de forma iterativa incorporar nuevos árboles al conjunto, pero cuando no lo consigue un predeterminado número de veces ($Max\_intento$), la ejecución se detiene. El pseudocódigo del algoritmo se presenta en Algoritmo 3.

Una vez que se ha inducido el conjunto de clasificadores, podemos usarlo para predecir. Hemos definido tres tipos de votación: uniforme, ponderada por árbol y ponderada por rama. La votación uniforme es la forma más sencilla: cada árbol da su vector de predicción y todos tienen el mismo peso. Para usar la votación ponderada hace falta un criterio que asigne pesos. En la votación ponderada por árboles, cada vector de predicción se pondera con el índice de acierto del árbol correspondiente. Para la votación ponderada por rama, el peso para ponderar el vector de predicción lo determina el peso de la rama dentro del árbol correspondiente.

Antes de pasar al apartado de experimentación comentaremos cómo afectan los parámetros a la ejecución del algoritmo y cuáles son los parámetros por defecto elegidos para su ejecución. Después de realizar algunos estudios empíricos con distintas configuraciones hemos observado lo siguiente:

- los mejores resultados se obtienen con la votación uniforme y la votación ponderada por árbol. Aún así, aunque existen diferencias entre los métodos de votación, éstas no son significativas. Por lo tanto, se ha elegido la votación uniforme como el método de votación por defecto por ser el más simple.

- la precisión suele ser mejor cuando en el conjunto hay más árboles. Además el tamaño medio de los árboles en el conjunto también es mejor cuando el conjunto está compuesto por más árboles. Para el parámetro del número máximo de árboles en el multiclasificador (Max_Arb) hemos elegido el valor 20.

- una vez que hemos fijado el tipo de votación y el número máximo de árboles, sólo nos resta fijar el número de intentos fallidos antes de detener la ejecución. Los multiclasificadores más precisos y cuyo tamaño de árbol medio es menor, se consiguen cuando se permiten pocos intentos fallidos. Para el parámetro del número de intentos fallidos ($Max\_intentos$) hemos elegido el valor 5.

4. Experimentación

En esta sección exponemos los resultados de algunos experimentos realizados. Los conjuntos de datos elegidos (Ecoli, Ionosphere, Pima Indians Diabetes y Wisconsin Breast Cancer) son del repositorio de la UCI (*UCI Machine Learning Repository* [1]) y sus características las mostramos en el Cuadro 1.

Cuadro 1: Conjuntos de datos usados para hacer los experimentos. Se dan el número de experiencias en el conjunto de datos, el número de atributos y el número de clases.

	Ecoli	Ionos.	Pima	Wdbc
Experienc.	336	351	768	569
Atributos	7	34	8	30
Clases	8	2	2	2

Todos los atributos de los conjuntos de datos que se han estudiado son continuos. Para poder trabajar con ellos los hemos discretizado previamente. Los datos que se muestran corresponden a las medias y desviaciones típicas de diez validaciones cruzadas con diez particiones para los algoritmos C4.5 (utilizamos la implementación hecha en Weka [15], que se llama J48), CIDIM, M-CIDIM y E-CIDIM. Se ha hecho así para presentar cómo se comporta CIDIM respecto a otro algoritmo que induce árboles de decisión y que es bastante conocido (C4.5) y para evidenciar la mejora que supone incorporar CIDIM dentro de multiclasificadores.

Cuadro 2: Precisiones alcanzadas con C4.5, CIDIM, M-CIDIM con 10 árboles y E-CIDIM con un máximo de 10 árboles y 5 intentos fallidos como máximo.

Problema	**Algoritmo**	**Precisión**
Ecoli	C4.5	77.33 ± 0.90
	CIDIM	77.29 ± 0.80
	M-CIDIM-10	80.51 ± 0.86
	E-CIDIM-10	80.49 ± 0.89
Ionos.	C4.5	87.85 ± 0.51
	CIDIM	88.71 ± 1.64
	M-CIDIM-10	90.65 ± 0.69
	E-CIDIM-10	90.39 ± 0.85
Pima	C4.5	73.85 ± 0.49
	CIDIM	73.29 ± 0.74
	M-CIDIM-10	74.17 ± 0.59
	E-CIDIM-10	73.63 ± 0.59
Wdbc	C4.5	93.01 ± 0.55
	CIDIM	92.48 ± 1.05
	M-CIDIM-10	94.43 ± 0.71
	E-CIDIM-10	95.15 ± 0.48

En lo que se refiere a la precisión podemos decir, después de ver el Cuadro 2, que la precisión alcanzada por CIDIM es comparable con la precisión alcanzada por C4.5. Pero la mejor precisión se alcanza con los multiclasificadores, tanto M-CIDIM como E-CIDIM, como era de esperar. Con los experimentos realizados no podemos determinar cuál de los dos multiclasificadores se comporta mejor.

En cuanto al tamaño del árbol (ver Cuadro 3, las comparaciones debemos realizarlas entre algoritmos semejantes, es decir, entre CIDIM y C4.5 o entre M-CIDIM y E-CIDIM. Como se puede apreciar, los árboles generados con CIDIM son mucho más pequeños que los generados con C4.5 con la consiguiente mejora

Cuadro 3: Tamaños de los árboles inducidos por C4.5, CIDIM, M-CIDIM con 10 árboles y E-CIDIM con un máximo de 10 árboles y 5 intentos fallidos como máximo.

Problema	Algoritmo	Hojas
Ecoli	C4.5	49.00 ± 1.52
	CIDIM	30.16 ± 1.64
	M-CIDIM-10	30.66 ± 0.53
	E-CIDIM-10	35.16 ± 0.25
Ionos.	C4.5	31.78 ± 1.13
	CIDIM	16.91 ± 1.07
	M-CIDIM-10	16.83 ± 0.26
	E-CIDIM-10	19.74 ± 0.35
Pima	C4.5	48.04 ± 5.34
	CIDIM	21.81 ± 2.33
	M-CIDIM-10	23.03 ± 1.44
	E-CIDIM-10	35.23 ± 1.19
Wdbc	C4.5	56.98 ± 2.30
	CIDIM	15.14 ± 1.58
	M-CIDIM-10	15.75 ± 0.55
	E-CIDIM-10	19.55 ± 0.35

en la comprensibilidad de los modelos. Las diferencias entre los multiclasificadores no son tan amplias, aunque parece que M-CIDIM genera multiclasificadores cuyo tamaño medio es menor.

5. Conclusiones y trabajos futuros

Este artículo presenta CIDIM, un algoritmo enfocado a la inducción de árboles de decisión precisos y pequeños. Hemos comparado los resultados obtenidos con C4.5 y observamos que las precisiones alcanzadas por ambos son muy similares, pero la aportación hecha por CIDIM que más destaca es que los árboles inducidos por él son más pequeños sin pérdida de precisión. Los algoritmos multiclasificadores basados en CIDIM que presentamos en este artículo también dan buenos resultados, mejorando la precisión y construyendo el conjunto de clasificadores con árboles pequeños.

Como posibles trabajos futuros tenemos los siguientes:

- realizar las modificaciones necesarias para que CIDIM pueda tratar directamente y de forma automática con atributos continuos y no haga falta discretizarlos previamente.
- mejorar los multiclasificadores proponiendo nuevas formas de votación ponderada. Esos mismos pesos usados para la ponderación podrían ser usados para hacer ajustes dentro del propio multiclasificador.
- automatizar la selección de los parámetros para el E-CIDIM de forma que no sea necesaria su configuración previa.
- integrar las características presentadas en M-CIDIM y E-CIDIM en un mismo algoritmo que se beneficie de las ventajas de cada uno de ellos.

Referencias

[1] C. Blake and C. J. Merz. UCI repository of machine learning databases. University of California, Department of Information and Computer Science, 2000.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[3] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00)*, pages 71–80. ACM Press, 2000.

[4] G. Folino, C. Pizzuti, and G. Spezzano. A cellular genetic programming approach to classification. In *Proceedings of the genetic and evolutionary computation conference*, volume 2, pages 1015– 1020. Morgan Kaufmann Publishers Inc., 1999.

[5] G. E. Naumov. NP-completeness of problems of construction of optimal decision trees. *Soviet Physics. Doklady*, 36:270–271, 1991.

[6] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Loh. Boat – optimistic decision

tree construction. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 169– 180. ACM Press, 1999.

[7] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5:15–17, 1976.

[8] J. M. Jerez-Aragonés, J. A. Gómez-Ruiz, G. Ramos-Jiménez, J. Muñoz Pérez, and E. Alba-Conejo. A combined neural network and decision trees model for prognosis of breast cancer relapse. *Artificial Intelligence in Medicine*, 27(1):45–63, 2003.

[9] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81– 106, 1986.

[10] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[11] G. Ramos-Jiménez, J. Campo-Ávila, and R. Morales-Bueno. Induction of decision trees using an internal control of induction. In *Proceedings of the International Work-Conference on Artificial Neural Networks (IWANN'2005) (to appear)*, 2005.

[12] G. Ramos-Jiménez, R. Morales-Bueno, and A. Villalba-Soria. CIDIM. Control of induction by sample division methods. In *Proceedings of the International Conference on Artificial Intelligence (IC-AI'00)*, pages 1083–1087, Las Vegas, June 2000.

[13] J. Ruiz-Gómez, G. Ramos-Jiménez, and A. Villalba-Soria. *Computers and Computacional Engineering in Control*, chapter Modelling based on rule induction learning, pages 158–163. World Scientific and Engineering Society Press, Greece, 1999.

[14] P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1):5– 44, 1997.

[15] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000.