ARTÍCULO

Sistemas multiclasificadores y de aprendizaje por capas basados en CIDIM

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

Departamento de Lenguajes y Ciencias de la Computación. E.T.S.I. Informática.
Universidad de Málaga. Málaga, 29071, España
{ramos,jcampo,morales}@lcc.uma.es

Resumen

En este artículo presentamos distintos sistemas multiclasificadores basados en CIDIM. También describimos un enfoque de aprendizaje por capas que hemos particularizado usando el algoritmo CIDIM como base. CIDIM (Control de Inducción por DIvisión Muestral) es un algoritmo que ha sido desarrollado para inducir árboles de decisión precisos y pequeños y para conseguirlo, intenta reducir el sobreajuste usando un control de inducción local. Los sistemas multiclasificadores que presentamos (M-CIDIM y E-CIDIM) aprovechan ciertas características de CIDIM, pero los enfoques desarrollados se pueden extender a otros algoritmos que compartan esas mismas características. Del mismo modo, el enfoque basado en aprendizaje por capas que proponemos no es sólo aplicable a un multiclasificador basado en CIDIM, sino que también es extensible a otros sistemas multiclasificadores.

Palabras clave: aprendizaje automático, sistemas multiclasificadores, aprendizaje por capas, control de inducción, algoritmo 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. Se han propuesto muchos modelos para representar dicho conocimiento, pero uno de los más usados es el árbol de decisión (CART [2], ID3 [8], C4.5 [9], ITI [17] ...). De entre todas las características positivas que tiene, nos gustaría destacar su capacidad de dividir el espacio de experiencias y ajustar cada uno de ellos con distintos modelos, además de la comprensibilidad del modelo generado.

La evaluación de la calidad de los árboles de decisión suele hacerse atendiendo a la precisión que se consigue con ellos y a su tamaño [3]. El principal objetivo es inducir árboles de decisión que sean precisos, pero, es deseable que, al mismo tiempo, sean lo más pequeños posibles. Como bien se sabe, el problema de encontrar el árbol más pequeño que clasifique completamente un conjunto de experiencias es un problema $NP - completo$ [6, 4] y, por consiguiente, diseñar una buena estrategia de expansión para el árbol se convierte en algo fundamental.

En este artículo presentamos CIDIM [12], un algoritmo que induce árboles de decisión usando un control local para la inducción. Este 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. Debemos comentar que una implementación de CIDIM ha sido utilizada para resolver problemas reales como modelado de sistemas [14] o modelado para el pronóstico de recidiba del cáncer de mama [7].

Tomando CIDIM como base, describiremos dos algoritmos multiclasificadores que aprovechan sus cualidades: 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.

Como última aportación en este artículo, proponemos un esquema de aprendizaje por capas. Este enfoque usa como base un sistema multiclasificador mucho más sencillo que los otros dos que proponemos. Así, veremos las mejoras por separado.

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 en la Sección 3 y el esquema de aprendizaje por capas se propone en la Sección 4. Una vez presentados los métodos, pasaremos, en la Sección 5, a mostrar los resultados de algunos experimentos. Para terminar presentaremos las conclusiones y algunos trabajos futuros en la Sección 6.

2. CIDIM

CIDIM [12] (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 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 [8, 9]) 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, estos subconjuntos se dividen de forma adecuada atendiendo a los nuevos hijos que se acaben de generar. De esta forma, el 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).

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

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

Figura 1. Subconjuntos CNS y CLS

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 quedan sin usar. Al igual que hace el algoritmo BOAT [5], 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. Para simplificar la configuración por defecto hemos usado la entropía para ambas medidas de desorden.

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 expandir. 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:

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

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

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 $P_{C(e)}(e) = \text{máx}\{P_1(e), P_2(e), ..., P_k(e)\}$ 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 (Figura 2) mostramos el pseudocódigo del algoritmo.

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).

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:
$Raiz$

Figura 2. Algoritmo CIDIM

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