Filtrando atributos para mejorar procesos de aprendizaje*

José Ramón Quevedo Pérez[H], Juan José del Coz Velasco[H], Jorge Díez Peláez[I]

[H]Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
Campus de Viesques S/N
33271 – Gijón
{ quevedo, juanjo }@aic.uniovi.es

[I]CENSYRA-SERIDA
C/ Camino de los Claveles 604
33271 – Gijón
jdiez@aic.uniovi.es

Resumen: *Los sistemas de aprendizaje automático han sido tradicionalmente usados para extraer conocimiento a partir de conjuntos de ejemplos descritos mediante atributos. Cuando la información de partida representa un problema real no se sabe, generalmente, qué atributos influyen en su resolución. En esos casos, la única opción a priori es utilizar toda la información disponible. Para evitar los problemas que esto conlleva se puede emplear un filtrado de atributos, previo al aprendizaje, que nos permita quedarnos sólo con los atributos más relevantes, aquellos que encierran la solución del problema. En este artículo se describe un método que realiza esta selección. Como se mostrará, está técnica mejora los procesos posteriores de aprendizaje.*

Palabras clave: Aprendizaje automático, relevancia y selección de atributos.

1 Introducción

Los sistemas de aprendizaje automático a partir de ejemplos tratan de inferir conocimiento de un conjunto de datos representado mediante una matriz donde cada fila corresponde a un ejemplo y las columnas contienen los atributos que lo describen, siendo la última de ellas la *categoría* que se pretende predecir. Esta categoría puede ser de dos tipos: una clase, en problemas simbólicos, o, como en los abordados en este artículo, una valoración numérica cuando el problema de predicción es de naturaleza continua.

* La investigación descrita en este artículo ha sido subvencionada por los proyectos PB98-1556 de la Dirección General de Enseñanza Superior e Investigación Científica y 1FD97-1633 de la CICYT (FEDER).

Los sistemas de aprendizaje tratan de encontrar las relaciones que hay entre el valor de los atributos en los ejemplos y las categorías de los mismos.

En muchos problemas de aplicación práctica la principal dificultad que se tiene para utilizar este proceso de aprendizaje es que no se sabe qué atributos de los que describen los ejemplos pueden tener relación con su categoría. En este caso la solución por la que se suele optar es incluir *todos* los atributos disponibles con la esperanza de que el sistema de aprendizaje sea capaz de descubrir los atributos *más relevantes* y considerarlos especialmente a la hora de aprender a distinguir las categorías. Este método tiene dos problemas importantes. El primero es que no se puede asegurar que la precisión de un sistema de aprendizaje no varíe al utilizar más atributos de los necesarios, es decir, al incluir en el conjunto de entrenamiento atributos irrelevantes para el problema, atributos que no tendrá ninguna relación con las categorías y que únicamente aportarán ruido al aprendizaje. En segundo lugar, cuando el número de atributos y ejemplos es muy elevado, los recursos necesarios (de tiempo y memoria) pueden hacer inviable el proceso de aprendizaje. Esta situación suele darse a menudo en casos prácticos donde se dispone de gran cantidad de información no organizada de la que se quiere extraer conocimiento.

En este artículo se muestra un método que evita tener que aprender con gran cantidad de atributos cuando sólo unos pocos son *necesarios* para la resolución del problema. Este método consiste en filtrar previamente el conjunto y obtener otro con el mismo número de ejemplos, pero únicamente con los atributos útiles; un sistema de aprendizaje empleará este nuevo conjunto y obtendrá el conocimiento buscado, utilizando menos recursos y posiblemente mejorando la calidad del aprendizaje realizado (Figura 1).

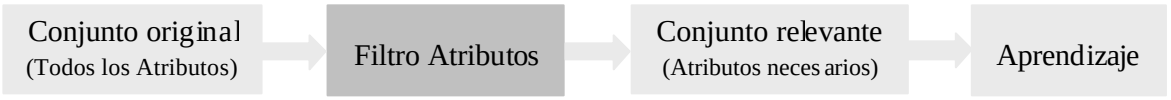

Figura 1. Mecanismo de filtrado de atributos para mejorar procesos de aprendizaje.

En los siguientes epígrafes se describirá en que consiste el método desarrollado para seleccionar atributos, se mostrarán los resultados de las pruebas experimentales efectuadas con esta nueva técnica sobre algunos conjuntos conocidos y otro de aplicación práctica y se indicarán las conclusiones que pueden extraerse de este trabajo.

2 Descripción general

El objetivo último de nuestro sistema es seleccionar el mejor subconjunto de atributos del conjunto de ejemplos original. Obviamente, el método óptimo consistiría en evaluar todos los subconjuntos de atributos posibles. Lo que ocurre es que esta solución nos lleva a una explosión combinatoria que impide que el sistema sea computacionalmente realizable, salvo que esté guiado mediante algún tipo de heurístico [BL97] [KJ97]. Por

lo tanto, la primera decisión que hay que tomar es buscar un método para hallar el mejor subconjunto de atributos, sin necesitar evaluar todas las posibilidades existentes.

La solución por la que hemos optado para evitar este escollo es efectuar una *ordenación de los atributos* que nos permita realizar un proceso guiado de eliminación. El objetivo de la ordenación debe ser tal que entre los subconjuntos de atributos generados y evaluados esté aquél del que pueda extraerse un conocimiento que ofrezca una mayor precisión. Para intentar lograrlo ordenaremos los atributos de menor a mayor relevancia y, mediante un proceso iterativo, buscaremos el *nivel óptimo* de relevancia para el problema. Es decir, partiendo del conjunto original, en cada iteración estimaremos la capacidad de precisión que puede obtenerse con el subconjunto de atributos resultante de eliminar del subconjunto de la iteración anterior el atributo menos relevante. De entre todos los subconjuntos probados nos quedaremos con aquél del que estimemos puede extraerse un conocimiento más exacto del problema.

El segundo elemento que precisamos, aunque éste más fácilmente resoluble, consiste en determinar cómo evaluar la capacidad de precisión que puede lograrse con cada uno de los subconjuntos de atributos testados. De entre las múltiples alternativas existentes escogeremos una buscando un compromiso entre la eficiencia y la estimación del error que ofrezca. El algoritmo final de nuestro sistema es el que sigue:

```
algoritmo FA (Filtrando Atributos) es
   MejorError=EstimacionError(ConjuntoOriginal)
   ConjuntoOrdenado=OrdenacionAtributos(ConjuntoOriginal)
   MejorSubconjunto=Subconjunto=ConjuntoOrdenado
   Para cada atributo A del ConjuntoOrdenado desde el peor atributo al mejor hacer
      Subconjunto=Subconjunto eliminado el atributo A
      ErrorSubconjunto=EstimacionError(Subconjunto)
      Si (ErrorSubconjunto < MejorError) entonces
            MejorError=ErrorSubconjunto
            MejorSubconjunto=Subconjunto
      finSi
   finPara
  Devolver MejorSubconjunto
FinAlgoritmo
```

3 Ordenación de atributos

Para resolver el problema de la ordenación de atributos, hemos optado por implementar dos métodos distintos: un mecanismo que ordena los atributos basándose en un sistema que asigna un peso a cada atributo y un método que determina la ordenación de acuerdo

al acierto que se obtiene con un algoritmo del vecino más próximo al eliminar del conjunto original cada uno de los atributos que lo forman.

3.1 Usando pesos para ordenar atributos

El primer método para ordenar los atributos de acuerdo a su importancia se basa en la técnica que emplea el sistema de aprendizaje BETS [Coz00] para determinar la relevancia. Esta técnica, al igual que otras existentes [AHA98] [WAM97], asigna un peso en el intervalo [0,1] a cada atributo. Para obtener cada uno de esos pesos, BETS estudia la capacidad de clasificación de las distintas ternas formadas por un atributo, un valor[1] dentro de ese atributo y la categoría que concluye. Es decir, la relevancia depende de lo discriminante que sea el valor en un atributo para concluir cada categoría continua. Para cada una de las ternas (atributo, valor, categoría) presentes en el conjunto de entrenamiento se calculará su poder de clasificación, para posteriormente comparar todos ellos y normalizar los datos obtenidos en el intervalo [0,1]. El valor normalizado calculado mediante este proceso es el peso que se otorgará a cada uno de los atributos de cada uno de los ejemplos de partida.

El poder de clasificación de cada terna se determina en función de su *Nivel de Impureza* [RMB97]. El nivel de impureza es una medida que trata de reflejar la calidad de clasificación de una regla. Lo que queremos medir es la calidad de las reglas del tipo:

$$categoría \Leftarrow atributo = valor \qquad (1)$$

Es decir, cómo el valor de un atributo nos puede ayudar a concluir una categoría. El cálculo de estos niveles de impureza es el que sigue. Por cada valor *v* en cada atributo *a*, se contará cuantas veces aparece (*n*) y que proporción (*p*) de esas veces concluye la categoría *c*. Con estos datos se calculará el *intervalo de confianza* de *p*, o lo que es lo mismo, el intervalo de confianza de la categoría *c* en el atributo *a* y para el valor *v*, [*EI*, *ED*], a partir de la fórmula:

$$\frac{p + \frac{z^2}{2n} \pm z\sqrt{\frac{p(1-p)}{n} + \frac{z^2}{4n^2}}}{1 + z^2/n} \qquad (2)$$

donde *z* se puede encontrar en la tabla de distribución normal de acuerdo al nivel de confianza deseado. El nivel de impureza mide el grado de solapamiento del intervalo de confianza de la regla anterior y el intervalo de confianza, [*EIAzar*, *EDAzar*], de la regla canónica del azar para la categoría *c* (una regla sin antecedentes que concluye en *c*).

[1] En caso de que el valor del atributo sea numérico, éste ha de entenderse como un intervalo resultante de discretizar el atributo. Lo mismo ocurre con la categoría, que en nuestros problemas será siempre continua.

$$NivelImpureza(a,v,c) = 100 \times \frac{EDAzar - EI}{ED - EI} \quad (3)$$

Una vez calculado el nivel de impureza de cada terna, procederemos a normalizarlos entre 1 y 0, tomando como 1 el mayor nivel de impureza medido y como 0 el valor 100. El valor 100 indica un nivel de relevancia igual que el azar, es decir, una relevancia nula. El cálculo definitivo queda así:

$$peso(a,v,c) = \frac{NivelImpureza(a,v,c) - 100}{MejorNivelImpureza - 100} \quad (4)$$

Una vez hallados los pesos para todas las ternas, se asignará en cada atributo de cada ejemplo el peso que le corresponda en función de los valores que contenga. Para ordenar los atributos del problema simplemente calcularemos el peso medio de todos ellos a través de todo el conjunto de entrenamiento. Aquellos atributos con un peso medio mayor se entiende que son más relevantes para la resolución del problema.

3.2 k-NN como método de ordenación de atributos

El sistema de aprendizaje de los *k* vecinos más próximos, o k-NN [CH67] [Das91], para evaluar un nuevo caso *e* calcula sus *k* vecinos (v_i), es decir, los *k* ejemplos del conjunto de entrenamiento más próximos a dicho caso, utilizando para ello la métrica euclídea donde los atributos continuos están normalizados y la distancia entre los simbólicos es 0 si es el mismo símbolo y 1 si es diferente.

En la versión de k-NN para problemas de categorías continuas, el número que se le otorgará será la media de las valoraciones de los ejemplos más próximos ponderándolos por el inverso de la distancia:

$$val(e) = \frac{\sum_{i=1}^{k} \frac{val(v_i)}{d(e,v_i)}}{\sum_{i=1}^{k} \frac{1}{d(e,v_i)}} \quad (6)$$

Si alguno de los *k* vecinos se encontrará a distancia 0, es decir, fuera exactamente igual al propio ejemplo *e*, a éste se le asignará directamente la valoración de aquél.

Un problema de este sistema de aprendizaje, como ocurre con otras muchas métricas y otros sistemas, es que no discrimina los atributos de acuerdo a su relevancia; según el método descrito anteriormente todos los atributos tienen la misma importancia a la hora de calcular la distancia entre dos ejemplos. La idea de este sistema se basa en que los ejemplos cercanos tienen categorías parecidas, pero esto solamente se cumple si los atributos (los cuales determinan si dos ejemplos son cercanos o no) tienen relación con dicha categoría. Si existen atributos que carecen de esa relación pueden determinar que dos ejemplos sean cercanos aún cuando sus categorías sean muy diferentes.

Teniendo en cuenta lo expuesto anteriormente, si a un conjunto de ejemplos le añadimos un atributo no relacionado con la categoría, el error de aprendizaje usando k-NN empeorará [CF94]. O lo que es lo mismo: "*si un conjunto de ejemplos tiene un atributo no relacionado con las categorías, al quitar este atributo, el error cometido por el aprendizaje usando k-NN disminuirá*". El método para determinar la relevancia de cada atributo se basa en esta afirmación. Consistirá, por tanto, en medir la precisión que se obtiene al eliminar cada atributo del conjunto original. Cuanto más relevante sea el atributo, en mayor medida disminuye la precisión que se alcanza con el resto de atributos.

4 Selección del mejor subconjunto de atributos

Siguiendo el principio expresado en el párrafo anterior, para medir la calidad de cada subconjunto de atributos estimaremos el error que k-NN comete con cada uno de los subconjuntos generados. Para que esta estimación resulte lo más precisa posible hemos utilizado el método *leaving-one-out*. Este método ofrece la estimación más fiable del error y consiste en el siguiente proceso: para cada ejemplo del conjunto se entrena el sistema con el resto de ejemplos y se calcula el error cometido; el error total será la media del error repitiendo este proceso con todos los ejemplos, es decir, excluyendo del entrenamiento una vez cada uno de ellos. Puede pensarse que esta estrategia es computacionalmente muy costosa, sin embargo, no debemos olvidar que el sistema utilizado es k-NN que no tiene coste de entrenamiento.

Por tanto, nuestra medida para evaluar la calidad de cada subconjunto de atributos será el error que cometa k-NN en *leaving-one-out* sobre dicho subconjunto. El mejor de todos ellos, será aquél para el que se obtenga un error menor.

5 Filtrado agresivo o prudente

La estrategia que se ha expuesto hasta este momento ha sido la de eliminar los atributos irrelevantes del problema, es decir, nos hemos limitado a realizar un *filtrado prudente*. Sin embargo, en determinadas circunstancias es posible que se desee algo más.

Un sistema de aprendizaje da mejores resultados cuando aprende a partir de *todos* los atributos relevantes del problema que aborda, pero, ¿qué sucede cuando aprende sólo a partir de los *más* relevantes, descartando algún atributo, que siendo relevante, lo es menos que los elegidos? Al descartar alguno de esos atributos relevantes parece lógico pensar que el sistema de aprendizaje pierda algo de precisión. Sin embargo, las ventajas que se pueden obtener a cambio pueden compensar esa pérdida.

Supongamos que la obtención de los ejemplos de entrenamiento es una tarea costosa y en la que el coste depende directamente del número de atributos que haya que medir,

algo muy habitual en problemas reales. En situaciones de ese tipo, es interesante reducir el número de atributos empleados lo más posible siempre que ello no conlleve una pérdida excesiva en la capacidad de precisión del sistema. Además, cuando eso se consigue, se logra que las explicaciones aportadas por el sistema de aprendizaje sean más comprensibles, en este caso se ha de resolver el problema con un *filtrado agresivo*.

El filtrado agresivo consiste en eliminar todos los atributos irrelevantes y algunos de los relevantes. El número de atributos relevantes eliminados puede variar dependiendo de las necesidades del problema. Por esto, en el proceso descrito anteriormente en el algoritmo del sistema, se hace necesario incluir un parámetro que indique el grado de agresividad del filtrado. Este parámetro se introduce como un porcentaje de forma que para considerar un nuevo subconjunto como el mejor, debe superar en un cierto porcentaje el mejor error registrado hasta ese momento. Actualmente se está utilizando la técnica del filtrado agresivo en un proyecto de investigación, la clasificación de canales bovinas cuyos resultados se mostrarán en el siguiente apartado.

6 Pruebas experimentales

El objetivo de estas pruebas se centró en el estudio de dos aspectos: la reducción del número de atributos en los conjuntos de ejemplos y la posible mejora en la capacidad de acierto al eliminar los atributos menos relevantes. Para calcular ambos elementos, se realizaron pruebas de validación cruzada con 10 particiones y 5 repeticiones; los resultados mostrados en todo este apartado corresponden a la media de los obtenidos en 50 aprendizajes. Para que estos experimentos sean repetibles se usó el sistema MLC++ [KJL94] con una semilla fija: 2032. Como sistemas de aprendizaje se emplearon: SAFE [Que00], M5' [WW97] y BETS [Coz00]; todos ellos aprenden categorías continuas.

En primer lugar, se probó la bondad del filtrado de atributos (FA) en conjuntos de ejemplos conocidos, obtenidos del repositorio de la UCI [BKM98]. Para asegurar la presencia de atributos irrelevantes, a estos conjuntos se les añadió pares de atributos aleatorios (uno continuo y otro simbólico), creando una serie de conjuntos con 0, 1, 2 y 3 pares de atributos irrelevantes. Para cada conjunto de la serie se calculó su error; la media de todos ellos es el error que se asignó a la serie generada a partir del conjunto original. Se calcularon los errores de los sistemas de aprendizaje por si solos y empleando previamente el filtrado de atributos. Como mecanismos de ordenación se utilizaron tanto los pesos de BETS como la ordenación obtenida gracias a k-NN. Para cada sistema se determinó el porcentaje de mejora, que es positivo si al utilizar FA mejora el error y negativo en otro caso. Todo ello aparece resumido en la Tabla I.

Con estas pruebas se consigue comparar el sistema de filtrado de atributos propuesto(FA) con respecto a los que implementan los sistemas de aprendizaje que se utilizan.

Tabla I. Porcentajes de mejora en la capacidad de acierto y reducción del número de atributos en problemas de categorías continuas conocidos. Cada problema representa una serie de conjuntos que resultan de añadir 0, 1, 2, 3 pares de atributos irrelevantes a los conjuntos originales[2]. Se compara la ordenación de atributos proporciona por los pesos de BETS y la obtenida con k-NN.

	k-NN FA					BETS FA				
	Tamaño	M5'	SAFE	BETS	*Media*	Tamaño	M5'	SAFE	BETS	*Media*
Cpu	64.2%	-0.7%	9.0%	14.7%	*7.7%*	55.8%	1.2%	11.5%	0.1%	*4.3%*
Autos	68.7%	0.3%	-0.5%	7.3%	*2.4%*	36.5%	3.9%	2.7%	6.8%	*4.5%*
Liver-disor.	75.3%	0.1%	-0.5%	1.9%	*0.5%*	66.5%	-1.8%	-0.5%	-2.6%	*-1.6%*
Meta	76.7%	-5.2%	-0.1%	-0,6%	*-2.0%*	17,1%	-32.0%	-3.4%	0.7%	*-11.6%*
Flags/pop	81.9%	-0.5%	0.9%	-2.5%	*-0.7%*	70.7%	-0.1%	-1.0%	-0.9%	*-0.7%*
Boston-hous.	71.2%	2.0%	0.0%	15,3%	*5.7%*	55,6%	-3.2%	-4.4%	1.9%	*-1.9%*

Usando la selección de atributos se consigue reducir el tamaño de los conjuntos de ejemplos, mientras que, en general, la precisión mejora o se mantiene, excepto en la serie del problema *meta* en la que el error aumenta notablemente, en gran parte por el error de M5'. Utilizando la ordenación de k-NN se consigue un filtrado más prudente, lo que conlleva un tamaño mayor y una precisión mejor; con la ordenación de BETS se logra un filtrado más agresivo y una precisión algo peor. En cualquier caso, la relación entre reducción de atributos y mejora de precisión puede calificarse de satisfactoria.

Una vez comprobado que el mecanismo es efectivo en conjuntos conocidos, lo aplicamos sobre un problema real: la calificación de canales bovinas. La obtención de datos para la calificación de canales es un proceso que se inicia en el matadero sacando tres fotografías de la canal (interna, externa y lateral). Sobre estas fotos y mediante un software implementado a tal efecto [DAL01], se marcan sus puntos y curvas características: un total de 25 puntos y 5 perfiles de los que se obtienen, mediante técnicas de visión artificial, las 27 medidas de la canal que forman los atributos de cada ejemplo. La categoría es un valor continuo entre 0.75 y 6.25 [GBA01].

El conjunto original, formado por 243 ejemplos de entrenamiento, ha sido sometido a un filtrado prudente que ha eliminado 13 atributos. Posteriormente, se han efectuado filtrados más agresivos reduciéndose el número de atributos a 9, 6 y 5. En la Tabla II se muestran los resultados con los sistemas SAFE, BETS y M5' sobre todos estos conjuntos.

Como puede verse, aplicando un filtrado prudente los algoritmos de aprendizaje mejoran con respecto al conjunto inicial. Aplicando filtrados agresivos el error se mantiene, salvo en el conjunto con 6 atributos, pero el número de atributos ha disminuido significativamente. Centrándonos en el conjunto que emplea sólo 5 atributos, el error

[2] Pueden obtenerse los conjuntos de ejemplos y los resultados detallados de la experimentación en : ftp://sella.aic.uniovi.es/publications/Machine_Learning/FAMPA.tar.gz

incluso ha bajado, es más, en el caso del sistema BETS de manera importante. En términos de obtención de ejemplos, la reducción de atributos permite que solamente se necesiten 4 puntos, 3 perfiles y 2 fotografías, la tercera se hace innecesaria. Para este problema el filtrado agresivo es muy interesante, ya que manteniendo la precisión, el coste económico y temporal en la recogida de datos disminuye drásticamente.

Tabla II. Error absoluto medio obtenido al ir eliminando atributos en el problema de la calificación de canales. La última fila contiene la media de los errores de los sistemas sobre cada conjunto. En este caso, la ordenación empleada en el proceso de filtrado se basó en los pesos asignados por BETS

Atributos	**27**	**14**	**9**	**6**	**5**
SAFE	0,4341	0,4362	0,4263	0,4601	0,4486
BETS	0,4605	0,4567	0,4353	0,4301	0,4115
M5'	0,4334	0,4212	0,4164	0,4634	0,4507
MEDIA	0,4427	0,4380	0,4260	0,4512	0,4369

7 Conclusiones

En este artículo se ha presentado un mecanismo que permite filtrar los atributos de un problema mejorando procesos posteriores de aprendizaje. Se ha demostrando que está técnica es capaz de reducir el número de atributos significativamente sin que se merme la capacidad de acierto del conocimiento obtenido. El algoritmo resultante se ha probado con éxito en un problema real (la clasificación de las canales bovinas) y actualmente se está aplicando en un problema que trata de detectar la mamitis en vacas productoras de leche.

Referencias

[Aha98] Aha, D. W.: Feature weighting for lazy learning algorithms. In H. Liu & H. Motoda (Eds.) Feature Extraction, Construction and Selection: A Data Mining Perspective. Norwell MA: Kluwer. 1998.

[BKM98] Blake, C., Keogh, E., Merz, C. J.: UCI Repository of machine learning databases ftp://ftp.ics.uci.edu/pub/machine-learning-databases. Irvine, University of California, Department of Information and Computer Sciences, 1998.

[BL97] Blum, A.L., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence, 97:245-271. 1997.

[CF94] Caruana, R., Freitag. D.: How useful is relevance? Working notes of the AAAI Fall Symposium on Relevance, 25-29. AAAI Press. 1994.

[CH67] Cover, T. M., Hart, P.E.: Nearest Neighbour pattern classification. IEEE Transactions on Information Theory, 13(1),21-27. 1967.

[Coz00] del Coz, J. J.: BETS. Sistema de aprendizaje basado en la selección de ejemplos paradigmáticos. Tesis Doctoral. Universidad de Oviedo. 2000.

[DAL01] Díez, J., Alonso, J., López, S., Bahamonde, A., Goyache, F.: Una aplicación informática para la representación informática de la conformación de canales Bovinas. Información Técnica Económica Agraria (ITEA), Vol. Extra, nº 22, Tomo II, 550-552. 2001.

[Das91] Dasarathy, B. V.: Nearest Neighbor (NN) norms: NN pattern classification techniques. Los Alamitos, CA: IEEE Computer Society Press. 1991.

[GBA01] Goyache, F., Bahamonde, A., Alonso, J., López, S, del Coz, J.J., Quevedo, J.R., Ranilla, J., Luaces, O., Alvarez, I., Royo, L.J., Díez, J.: Usefulness of Artificial Intelligence techniques to assess subjective quality of products in food industry. Pendiente de publicación en Trends in Food Science and Technology (2001).

[KJ97] Kohavi, R., John, G.: Wrappers for feature subset selection. Artificial Intelligence, 97, 273-324. 1997.

[KJL94] Kohavi, R., John G., Long, R., Manley D., Pfleger K.: MLC++: A machine learning library in C++. In: Proceedings of the Sixth International Conference on Tools with Artificial Intelligence, 740-743. 1994.

[Que00] Quevedo, J.R.: SAFE : Sistema de aprendizaje de funciones a partir de ejemplos. Tesis Doctoral. Universidad de Oviedo. 2000.

[RMB97] Ranilla, J., Mones, R., Bahamonde, A.: El nivel de impureza de una regla de clasificación aprendida a partir de ejemplos. Actas de la VII Conferencia de la Asociación Española para la Inteligencia Artificial, 479-488. 1997.

[WAM97] Wettschereck, D., Aha D. W. y Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review, 11, 273-314. 1997.

[WW97] Wang, Y. Witten, I. H.: Inducing Model Trees for Predicting Continuous Classes. In Proceedings of European Conference on Machine Learning. University of Economics. Prague. 1997.