Mejorando el comportamiento de ensambles basados en boosting, bagging y random forest mediante Soft computing

José M. Cadenas y M. Carmen Garrido
Dpto. Ingeniería de la Información y las Comunicaciones. Universidad de Murcia
Murcia, 30071, España

y

Ramón A Díaz-Valladares
Área de postgrado e Investigación. Universidad de Montemorelos
Montemorelos, 67500, México

RESUMEN

En este trabajo, aplicamos el soft computing a los ensambles basados en boosting, bagging y random forest que usan como clasificador base un árbol de decisión. Definimos varios métodos de combinación para proporcionar la decisión final de estos ensambles. Al introducir el soft computing en los ensambles basados en árboles de decisión pretendemos aprovechar la mejora que proporcionan las técnicas basadas en ensambles con respecto a los clasificadores individuales, incrementar la robustez al ruido y permitir el tratamiento de incertidumbre.

Palabras Clave: Soft computing, árbol de decisión fuzzy, ensamble, método de combinación.

1 INTRODUCCIÓN

Con el objetivo de mejorar la precisión de las predicciones, ha surgido un interés creciente en los últimos años en la definición de métodos que combinan hipótesis. Estos métodos construyen un conjunto de hipótesis (ensamble), y combinan las predicciones del conjunto de alguna manera (normalmente por votación) para clasificar nuevos ejemplos. La precisión obtenida por esta combinación de hipótesis supera, generalmente, la precisión de cada componente individual del conjunto [1]. A estas técnicas de combinación de hipótesis se les denomina métodos multiclasificadores o métodos de ensamble [7].

Se han propuesto numerosos métodos de ensamble donde cada componente individual se construye usando el mismo método clasificador, denominado clasificador base, modificando el conjunto de datos de aprendizaje de cada uno de ellos, o el conjunto de atributos del conjunto de datos de aprendizaje o introduciendo algún factor aleatorio en el proceso de construcción del clasificador [7]. En este trabajo nos centramos en ensambles que utilizan como clasificador base un árbol de decisión.

Sin embargo, aunque los árboles de decisión son interpretables, eficientes y capaces de trabajar con bases de datos grandes, son muy inestables cuando pequeñas cantidades de ruido se introducen en los datos de aprendizaje. Por esta razón, el soft computing y la lógica fuzzy se han incorporado en las técnicas de contrucción de árboles de decisión. En [11, 12, 13] podemos encontrar propuestas en las que la teoría de conjuntos fuzzy y su capacidad de razonamiento aproximado se ha combinado con la técnica de construcción de árboles de decisión. Esta integración preserva las ventajas de ambos componentes: el tratamiento de la incertidumbre con la comprensibilidad de las variables linguísticas de la lógica fuzzy y la popularidad y fácil aplicación de los árboles de decisión. Los árboles resultantes muestran un incremento de su robustez ante el ruido y una aplicabilidad extendida a la incertidumbre o contextos vagos.

En este trabajo, consideramos los principales ensambles basados en árboles de decisión de la literatura y contruimos sus equivalentes usando como clasificador base un árbol de decisión fuzzy, realizando una comparación entre ambos grupos: ensambles basados en árboles de decisión crisp versus ensambles basados en árboles de decisión fuzzy. Con los ensambles basados en árboles de decisión fuzzy pretendemos aprovechar la mejora en resultados que proporcionan las técnicas basadas en ensambles con respecto a los clasificadores individuales, incrementar la robustez al ruido de los ensambles basados en árboles de decisión crisp y permitir el manejo de incertidumbre.

De este forma, en este trabajo presentamos en la sección 2 el árbol de decisión fuzzy que usaremos como clasificador base, mostrando sus fases de aprendizaje e inferencia. En la sección 3 definimos los distintos ensambles basados en árboles de decisión fuzzy indi-

cando sus fases de aprendizaje e inferencia. En la sección 4, aplicamos los distintos ensambles a una serie de bases de datos y, por último, mostramos las conclusiones obtenidas de nuestro trabajo.

2 UN ÁRBOL DE DECISIÓN FUZZY

En esta sección describimos el árbol fuzzy que va a actuar como clasificador base de los ensambles. En primer lugar mostraremos el algoritmo de aprendizaje del árbol a partir de un conjunto de ejemplos y despues pasaremos a describir el algoritmo de inferencia que permite predecir la clase desconocida de un ejemplo dado.

El conjunto de ejemplos E a partir del cual se construye el árbol, está formado por ejemplos descritos por atributos que pueden ser nominales y numéricos y donde al menos habrá un atributo nominal que actúa como atributo clase.

El algoritmo mediante el cual construimos el árbol fuzzy está basado en el algoritmo ID3 donde todos los atributos numéricos han sido discretizados mediante una partición fuzzy previamente obtenida mediante un algoritmo de discretización basado en un algoritmo genético [6]. La partición fuzzy de cada atributo numérico garantiza: la completitud (ningún valor del dominio queda fuera de la partición fuzzy), y es una partición fuzzy fuerte (satisface que $\forall x \in E$, $\sum_{i=1}^{f} \mu_{A_i}(x) = 1$, donde $A_1, \ldots, A_f$ son los conjuntos fuzzy de la partición). Por lo tanto, el dominio de cada atributo numérico es representado en el árbol mediante los conjuntos fuzzy trapezoidales $A_1, \ldots, A_f$.

A cada ejemplo e usado en el aprendizaje del árbol t se le asigna un valor inicial igual a 1 ($\chi_{t,raiz}(e) = 1$) indicando que inicialmente el ejemplo está sólo en el nodo raíz de t. Dicho valor seguirá siendo 1 mientras el ejemplo e no pertenezca a más de un nodo durante el proceso de construcción del árbol. En un árbol clásico, un ejemplo sólo puede pertenecer a un nodo en cada momento, por lo tanto, su valor inicial (si existiera) no se modificaría a lo largo del proceso de construcción. En el caso de un árbol fuzzy, este valor es modificado en dos situaciones:

- Cuando el ejemplo e tiene un valor missing en un atributo que se usa como test en un nodo. En este caso el ejemplo desciende a cada nodo hijo con un valor modificado como $\chi_{t,nodo\_hijo}(e) = \chi_{t,nodo}(e) \times \frac{1}{numero\_salidas_{test}}$.
- De acuerdo al grado de pertenencia de e a los distintos conjuntos fuzzy de la partición cuando el test en un nodo está basado en un atributo numérico. En este caso, el ejemplo desciende a aquellos nodos hijos a los que el ejemplo pertenezca con grado mayor de 0 ($\mu_{conjunto\_fuzzy\_particion}(e) > 0$). Por las características de las particiones que usamos, el ejemplo podrá descender como mucho a dos nodos hijos. En este caso, $\chi_{t,nodo\_hijo}(e) = \chi_{t,nodo}(e) \times \mu_{conjunto\_fuzzy\_particion}(e)$.

Podemos decir que el valor $\chi_{t,nodo}(e)$ indica el grado con el cual el ejemplo satisface las condiciones que conducen a nodo *nodo* del árbol t.

En el Algoritmo 1, se muestra el algoritmo de construcción de un árbol fuzzy a partir de un conjunto de ejemplos.

Algoritmo 1 - Aprendizaje de un árbol fuzzy

ArbolFuzzy(*in* : E, $Particion\ Fuzzy$;
out : $Arbol\ Fuzzy\ t$)

begin

1. Sea el nodo raíz constituido por los ejemplos de E con valores $\chi_{t,raiz}(e) = 1$.
2. Sea M el conjunto de atributos que describen los ejemplos de E y donde los atributos numéricos han sido discretizados según la $Particion\ Fuzzy$.
3. Seleccionar de M un atributo como test en el nodo N. Para ello:
   3.1. Calcular la ganancia de información para cada atributo en M usando los valores $\chi_{t,N}(e)$ de cada ejemplo e en el nodo N.
   3.2. Elegir el atributo con ganancia máxima.
4. Dividir N en subnodos de acuerdo a las posibles salidas del atributo seleccionado en el paso previo y eliminar dicho atributo de M. Sean $E_{N_1}, ..., E_{N_{numero\_salidas_{test}}}$, los conjuntos de datos en cada subnodo.
5. Repetir los pasos 3 y 4 para cada E_{N_i} y M con $i = 1, ..., numero\_salidas_{test}$ hasta que se cumpla la condición de parada.

end

Hay que destacar, como se indica en el paso 4 del algoritmo 1, que una vez seleccionado un atributo como test en un nodo, dicho atributo no volverá a ser seleccionado debido al hecho de que todos los atributos o son nominales o están particionados.

La condición de parada en el algoritmo 1 está definida por la primera condición alcanzada de las siguientes:

(1) nodo puro, (2) conjunto M vacío, (3) alcanzar el mínimo número de ejemplos permitido en un nodo.

Una vez construido el árbol lo usamos para inferir la clase desconocida de un nuevo ejemplo siguiendo el algoritmo 2.

Algoritmo 2 - Inferencia en un árbol fuzzy

ArbolFuzzy(*in* : *Arbol Fuzzy t*, *Particion Fuzzy*, *Ejemplo e*; *out* : *Clase c*)

begin

1. Dado el ejemplo e, que queremos clasificar, con valor inicial $\chi_{t,raiz}(e) = 1$, recorrer el árbol desde el nodo raíz.
2. Obtener el conjunto de hojas alcanzadas por e. e puede ramificarse por más de una rama si en algún test basado en un atributo numérico, e pertenece a más de un elemento de la partición de dicho atributo con grado mayor que 0. También se puede ramificar por más de una rama si tiene al menos un valor desconocido en los atributos usados por el árbol como test.
3. Para cada hoja alcanzada por e, calcular el soporte para cada clase. El soporte de una clase en una hoja dada n se obtiene sumando los valores $\chi_{t,n}(ejemplo)$ de todos los ejemplos que alcanzaron dicha hoja durante la fase de aprendizaje y que pertenecen a dicha clase.
4. Obtener la decisión del árbol c a partir de la información proporcionada por el conjunto de hojas alcanzadas N_t y el valor $\chi_{t,n}(e); n = 1, ..., |N_t|$ con el que el ejemplo activa o alcanza cada hoja del árbol.

end

Como se indica en el paso 4 del algoritmo 2, una característica de un árbol fuzzy frente a un árbol crisp es que un ejemplo, en su recorrido por el árbol, puede alcanzar o activar más de una hoja. Por lo tanto, al hacer la inferencia se pueden usar múltiples estrategias. Algunas de ellas son:

- Considerar solamente la información proporcionada por la hoja activada con mayor valor por el ejemplo a clasificar. En este caso, el árbol decide la clase con mayor soporte en esta hoja.
- Considerar todas las hojas alcanzadas por el ejemplo y obtener la decisión final del árbol en base al soporte proporcionado para cada clase por cada una de ellas y el valor $\chi_{t,n}(e)$ con el que el ejemplo e activa la hoja n, usando operadores de agregación como voto mayoritario, mínimo, máximo, promedio y versiones ponderadas de todos ellos.

3 ENSAMBLES BASADOS EN ÁRBOLES DE DECISIÓN FUZZY

En esta sección vamos a presentar cómo se construyen los principales ensambles basados en árboles que aparecen en la literatura a partir del algoritmo de construcción de un árbol fuzzy mostrado en la sección anterior (Algoritmo 1). Además presentaremos cómo realizar la inferencia con dichos ensambles.

3.1 Fase de aprendizaje

Nos vamos a centrar en los ensambles Bagging [4], Boosting [8, 14] y Random Forest [5], debido a que son los ensambles más utilizados en la literatura.

Bagging [4] es una de las técnicas más antigua, simple y bien conocida para crear un ensamble de clasificadores. En bagging, la diversidad se obtiene construyendo cada clasificador con un conjunto de ejemplos diferente que se obtienen seleccionando ejemplos con reemplazamiento del conjunto de ejemplos original. Por lo tanto, para construir un ensamble basado en bagging formamos diferentes conjuntos de ejemplos de la forma comentada anteriormente y le aplicamos a cada uno de ellos el algoritmo base que en nuestro caso es el algoritmo 1.

Boosting [8, 14] es un algoritmo que crea un ensamble de clasificadores añadiendo un clasificador en cada paso. En el conjunto de ejemplos original, cada ejemplo tiene asignado un peso. El clasificador que se añade en el paso k es aprendido a partir del conjunto de ejemplos original donde el peso de cada ejemplo ha sido modificado en la iteración $k-1$ del algoritmo. En cada iteración, boosting aprende un modelo que minimiza la suma de los pesos de los ejemplos clasificados erróneamente. Los errores en cada iteración se utilizan para actualizar los pesos de los ejemplos del conjunto de entrenamiento de manera que se incremente el peso de los ejemplos erróneos y se reduzca el peso de los ejemplos clasificados correctamente. Cuando usamos como algoritmo base el algoritmo 1 en un ensamble boosting, el peso de cada ejemplo e en la base de datos original será el valor $\chi_{t,raiz}(e)$ definido anteriormente y estará inicializado a 1.

Finalmente, Breiman [5] presentó el ensamble denominado random forest donde se utiliza bagging junto a una selección aleatoria de atributos. En cada nodo de cada árbol del bosque, se selecciona aleatoriamente un subconjunto de los atributos disponibles en ese nodo

y se elige el mejor de ellos de acuerdo al criterio de división empleado en el algoritmo base. El número de atributos seleccionado aleatoriamente es un parámetro de entrada y diversos estudios avalan que el valor $log_2(|M|-1)$, donde $|M|$ es el cardinal del conjunto de atributos disponibles en ese nodo en cuestión, produce un buen comportamiento del ensamble. Si usamos el Algoritmo 1 como algoritmo base de un ensamble basado en random forest debemos modificar el paso 3 de la siguiente forma:

Modificación Algoritmo 1

3. Seleccionar aleatoriamente de M un subconjunto de atributos denominamos RM. Seleccionar de RM un atributo (test) en el nodo N. Para ello:
   3.1. Calcular la ganancia de información para cada atributo de RM usando los valores $\chi_{t,N}(e)$ de cada ejemplo e en el nodo N.
   3.2. Elegir el atributo con ganancia máxima.

Además, dado que el ensamble random forest hace uso de bagging, el conjunto de entrenamiento de cada clasificador se forma seleccionando ejemplos con reemplazamiento a partir del conjunto de ejemplos original. Por lo tanto, un ensamble random forest está basado en un doble nivel de aleatoriedad para conseguir diversidad.

3.2 Fase de inferencia

En esta sección vamos a describir cómo realizar la inferencia o clasificación de un nuevo ejemplo una vez que el ensamble ha sido construido. Dado que los ensambles a los que nos referimos usan como clasificador base el árbol fuzzy del Algoritmo 1 vamos, en primer lugar, a comentar aquellos aspectos que se pueden modificar en la inferencia de los ensambles basados en árboles crisp debido al uso del árbol fuzzy como clasificador base.

Como comentamos al presentar el Algoritmo 2, una característica del árbol fuzzy es que un ejemplo a clasificar puede alcanzar o activar más de un nodo hoja. Este hecho nos permite poder tener en cuenta la decisión de cada hoja alcanzada de cada árbol del ensamble como veremos en la definición de posibles formas de inferencia que veremos a continuación. Por otro lado, las hojas son alcanzadas o activadas por el ejemplo e con un valor $\chi_{t,n}(e); n = 1, ..., |N_t|$. Este valor también puede ser considerado en la definición del proceso inferencia para obtener la decisión final del ensamble acerca de la clase. Además, en la decisión de cada hoja alcanzada se considera la suma de los valores $\chi_{t,n}(ejemplo)$ de los ejemplos que alcanzaron esa hoja durante la fase de aprendizaje en lugar del número de ejemplos en dicha hoja.

Teniendo en cuenta las consideraciones anteriores, en un ensamble basado en árboles fuzzy como clasificador base vamos a considerar dos estrategias generales de inferencia:

Estrategia 1: Consiste en agregar la información de las diferentes hojas alcanzadas de cada árbol para obtener la decisión individual de cada árbol y después agregar la información de cada árbol para generar la decisión global del ensamble. En la figura 1 mostramos gráficamente el esquema de la estrategia 1.

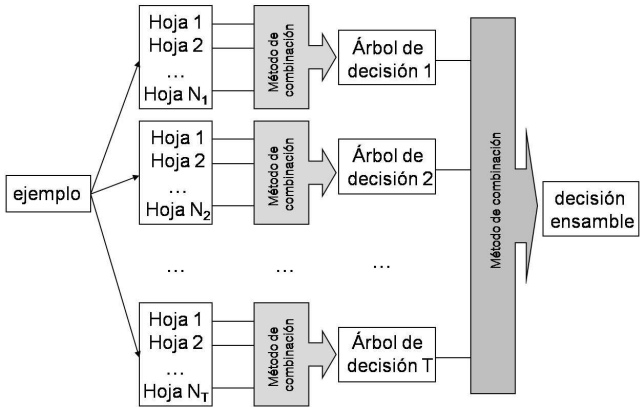

Figura 1: Esquema de la estrategia 1 para inferencia

Estrategia 2: Consiste en agregar la información de las hojas alcanzadas de cada árbol para generar la decisión del ensamble. Como hemo presentado en la estrategia 1, en la figura 2 mostramos gráficamente el esquema de la estrategia 2.

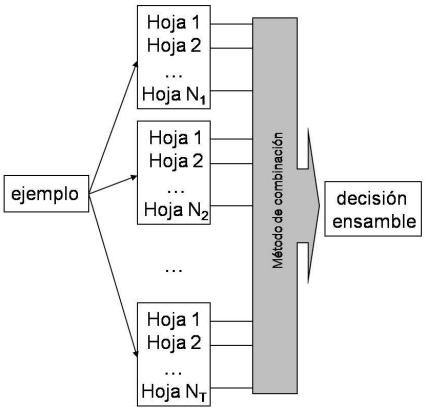

Figura 2: Esquema de la estrategia 2 para inferencia

Para definir métodos de combinación concretos para cada una de estas estrategias, vamos a considerar qué información podemos obtener de cada hoja alcanzada por el ejemplo e en un ensamble. Esta información puede ser:

- voto mayoritario: la hoja puede proporcionar voto 1 a la clase con mayor soporte y 0 para las demás. El voto de cada hoja tiene la misma importancia.
- voto mayoritario ponderado: la hoja alcanzada asigna un voto con valor $\chi_{t,n}(e)$ a la clase con mayor soporte y 0 para las demás. De esta forma, el grado de activación de cada hoja alcanzada por el ejemplo e es considerado en el voto de la hoja.

Teniendo en cuenta lo anterior, vamos a definir algunos métodos de combinación concretos:

- Voto mayoritario simple.
  - En la estrategia 1 (SM1):

    Cada hoja alcanzada por el ejemplo e en el ensamble proporciona su voto mayoritario. Estos votos son sumados por árbol, es decir, se suman para cada clase los votos de las hojas alcanzadas pertenecientes a un mismo árbol de forma que dicho árbol realiza un voto mayoritario por la clase más votada entre sus hojas alcanzadas. El ensamble decide la clase más votada por sus árboles.

  - En la estrategia 2 (SM2):

    Cada hoja alcanzada por el ejemplo e en el ensamble proporciona su voto mayoritario. El ensamble decide la clase más votada por las hojas alcanzadas.

- Voto mayoritario ponderado por el valor de activación de la hoja.
  - En la estrategia 1 (MWL1):

    Cada hoja alcanzada por el ejemplo e en el ensamble proporciona su voto mayoritario ponderado. Para cada clase, los valores de estos votos son sumados por árbol, de forma que dicho árbol realiza un voto mayoritario por la clase con mayor valor final. El ensamble decide la clase más votada por sus árboles.

  - En la estrategia 2 (MWL2):

    Cada hoja alcanzada por el ejemplo e en el ensamble proporciona su voto mayoritario ponderado. El ensamble decide la clase con mayor valor una vez sumados para cada clase los votos ponderados de cada hoja.

- Voto mayoritario ponderado por el valor de activación de la hoja y un peso por árbol.
  - En la estrategia 1 (MWLT1):

    En este método asignamos un peso a cada árbol del ensamble en función de la proporción de errores cometidos al clasificar un conjunto de ejemplos con dicho árbol. De esta forma, el método de combinación queda definido de forma que cada hoja alcanzada por el ejemplo e en el ensamble proporciona su voto mayoritario ponderado. Para cada clase, los valores de estos votos son sumados por árbol, de forma que dicho árbol vota con su peso por la clase con mayor valor final. El ensamble decide la clase con mayor valor una vez sumados para cada clase los votos de cada árbol.

  - En la estrategia 2 (MWLT2):

    De nuevo, en este método asignamos un peso a cada árbol del ensamble en función de la proporción de errores cometidos al clasificar un conjunto de ejemplos con dicho árbol. De esta forma, cada hoja alcanzada por el ejemplo e en el ensamble proporciona su voto mayoritario ponderado que en este caso considera el valor con el que fue activada multiplicado por el peso asignado al árbol al que pertenece. El ensamble decide la clase con mayor valor una vez sumados para cada clase los votos ponderados de cada hoja.

- Voto basado en una única hoja y ponderado por el peso del árbol para la estrategia 1 (WT1):

  En la decisión de cada árbol sólo se considera la hoja activada con mayor valor por el ejemplo e y el árbol asigna su voto, ponderado por el peso que tiene asignado como árbol, a la clase con mayor soporte en dicha hoja. El ensamble decide la clase con mayor valor una vez sumados para cada clase los votos ponderados de cada árbol.

En la siguiente sección, todos estos métodos de combinación serán aplicados a los ensambles basados en árboles fuzzy comentados anteriormente y compararemos sus resultados con los proporcionados por ensambles basados en árboles crisp de la literatura.

4 EXPERIMENTOS

En esta sección hemos seleccionado una serie de bases de datos de la UCI Machine Learning Repository [2] para obtener una serie de resultados experimentales sobre los distintos ensambles basados en árboles fuzzy que hemos mencionado en este trabajo. Además se han aplicado dichas bases de datos a las versiones crisp de los ensambles anteriores, es decir, ensambles que usan un árbol crisp como clasificador base.

La descripción de las bases de datos empleadas se muestra en la tabla 1.

Tabla 1: Descripción de las Bases de datos

Base de datos	Abbr	$\|E\|$	$\|M\|$	nº clases
Contraceptive Method	CMC	1473	9	3
Statlog Heart Disease	HEA	270	13	2
Ionosphere	ION	351	34	2
Iris Plants	IRP	150	4	3
Pima Indian Diabetes	PID	532	7	2
Attitude Smoking	SMO	2855	8	3

La descripción de los experimentos realizados es la siguiente:

- Para ejecutar los ensambles fuzzy hemos implementado el árbol fuzzy del algoritmo 1 y lo hemos integrado en las distintas técnicas basadas en ensambles.
- Para ejecutar los ensambles crisp hemos usado las implementaciones bagging, boosting y random forest que proporciona la herramienta Weka [10] usando el árbol J48 como clasificador base.
- La ejecución de cada experimento ha sido realizada con los mismos parámetros o equivalentes.
- Los resultados muestran el porcentaje medio de precisión (media y desviación estandar) de una validación cruzada de 4×5 (repetimos 4 veces una validación cruzada de tamaño 5).
- Una vez obtenidos los resultados de los experimentos, hemos realizado un análisis usando técnicas estadísticas. Utilizamos la metodología propuesta en [9] y usamos tests no paramétricos para realizar dicho análisis.

En la tabla 2 se muestran los resultados para los ensambles que usan un árbol crisp como clasificador base.

En la tabla 3 se muestran los resultados obtenidos con los ensambles basados en árbol fuzzy.

Pasamos a realizar el análisis estadístico de los resultados obtenidos. Para ello usamos el test de Friedman y el procedimiento de Benjamin-Hochberger [3] como test post-hoc.

Para analizar el rendimiento de los 6 ensambles y los 2 clasificadores base, hemos aplicado primero el test de Friedman. Se rechaza la hipótesis nula con un nivel de confianza del 95%, es decir, aceptamos que hay diferencias significativas entre los métodos. Cuando hacemos el análisis post-hoc, obtenemos que el ensamble Random Forest fuzzy tiene diferencias significativas con todos los demás métodos con un nivel de confianza del 95%, siendo el mejor de ellos. Además, obtenemos que Bagging fuzzy es mejor que el árbol fuzzy, Boosting y J48 con un nivel de confianza del 93.7% y mejor que Boosting fuzzy y Random Forest al 90.6%. Por último, obtenemos que Boosting fuzzy es mejor que el árbol fuzzy con un nivel de confianza del 92.15%.

De los métodos de combinación presentados en la sección anterior, los que obtienen los mejores resultados (en 13 de las 18 pruebas realizadas) son los denominados MWLT1 y MWLT2.

5 CONCLUSIONES

En este trabajo hemos presentado los principales ensambles de la literatura basados en árboles de decisión sustituyendo el árbol crisp por un árbol fuzzy como clasificador base de los ensambles. Hemos indicado los principales aspectos a modificar tanto en la construcción como en la inferencia de dichos ensambles, presentando diversos métodos de combinación para fusionar las salidas de los árboles del ensamble.

Hemos llevado a cabo la comparación de los ensambles basados en árboles crisp y los basados en árboles fuzzy realizando una serie de experimentos. Los resultados obtenidos para un conjunto de bases de datos muestran que los ensambles fuzzy obtienen siempre un mejor resultado que el clasificador individual fuzzy. Por otro lado, el ensamble fuzzy basado en random forest es el que consistentemente obtiene los mejores resultados y muestra el mejor comportamiento. Los métodos de combinación que obtienen los mejores resultados son los métodos basados en voto mayoritario ponderado por el valor de activación de la hoja y peso por árbol. Estos resultados han sido validados mediante la realización de un análisis estadístico.

Tabla 2: Ensambles basados en árboles crisp

BD	J48	Tamaño ensambles	Random Forest	Bagging	Boosting
CMC	$50.98_{(0.00)}$	120	$51.44_{(0.35)}$	$54.36_{(0.32)}$	$51.12_{(0.00)}$
HEA	$77.78_{(0.00)}$	120	$81.48_{(0.78)}$	$81.30_{(0.32)}$	$81.48_{(0.00)}$
ION	$89.74_{(0.00)}$	175	$93.38_{(0.24)}$	$92.24_{(0.13)}$	$94.59_{(0.00)}$
IRP	$96.00_{(0.00)}$	120	$95.67_{(0.34)}$	$96.00_{(0.00)}$	$92.67_{(0.00)}$
PID	$75.94_{(0.00)}$	50	$75.75_{(0.55)}$	$76.69_{(0.59)}$	$74.44_{(0.00)}$
SMO	$69.53_{(0.00)}$	75	$61.05_{(0.14)}$	$65.05_{(0.19)}$	$58.63_{(0.00)}$

Tabla 3: Ensambles basados en árboles fuzzy

BD	Árbol fuzzy	Tamaño ensambles	Random Forest fuzzy	Bagging fuzzy	Boosting fuzzy
CMC	$47.17_{(2.62)}$	120	$53.31_{(2.32)}$	$51.43_{(2.10)}$	$49.59_{(1.78)}$
HEA	$74.26_{(6.58)}$	120	$82.31_{(4.70)}$	$81.02_{(5.13)}$	$75.37_{(5.60)}$
ION	$92.59_{(3.29)}$	175	$94.66_{(2.19)}$	$93.95_{(3.12)}$	$94.59_{(2.25)}$
IRP	$97.00_{(2.08)}$	120	$97.33_{(2.14)}$	$96.67_{(2.53)}$	$97.50_{(2.60)}$
PID	$71.85_{(3.95)}$	50	$79.32_{(3.72)}$	$77.54_{(3.82)}$	$75.89_{(3.76)}$
SMO	$55.29_{(2.42)}$	75	$69.54_{(1.97)}$	$69.48_{(1.87)}$	$59.27_{(3.52)}$

Agradecimientos

Los autores agradecen al "Ministerio de Educación y Ciencia" y al "Fondo Europeo de Desarrollo Regional" por el soporte, bajo el proyecto TIN2008-06872-C04-03, para el desarrollo de este trabajo. Gracias también a la "Fundación Séneca" por el programa para Grupos de Investigación de Excelencia con código 04552/GERM/06.

6 REFERENCIAS

[1] H. Ahn, H. Moon, J. Fazzari, N. Lim, J. Chen, R. Kodell, "Classification by ensembles from random partitions of high dimensional data", **Computational Statistics and Data Analysis**, Vol. 51, 2007, pp. 6166–6179.

[2] A. Asuncion, D.J. Newman, **UCI Machine Lear-ning Repository**, *http://www.ics.uci.edu/ mlearn/ MLRepository.html*, Irvine, CA: University of California, 2007.

[3] Y. Benjamini, Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing", **Journal of the Royal Statistical Society Series B**, Vol. 57, 1995, pp. 289–300.

[4] L. Breiman, "Bagging predictors", **Machine Learning**, Vol. 24, No. 2, 1996, pp. 123–140.

[5] L. Breiman, "Random forests", **Machine Learning**, Vol. 45, No. 1, 2001, pp. 5–32.

[6] J.M. Cadenas, M.C. Garrido, R. Martínez, "Una estrategia de particionamiento fuzzy basada en combinación de algoritmos", **In proceedings XIII Conferencia de la Asociación Española para la Inteligencia Artificial**, 2009, pp. 379–388.

[7] T.G. Dietterich, " An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization", **Machine Learning**, Vol. 40, No. 2, 2000, pp. 139–157.

[8] Y. Freud, R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", **Journal of Computer and System Sciences**, Vol. 55, No. 1, 1997, pp. 119–139.

[9] S. García, A. Fernández, J. Luengo, F. Herrera, "A study statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability", **Soft Computing**, Vol. 13, No. 10, 2009, pp. 959–977.

[10] I.H. Witten, E. Frank, **Data Mining: Practical machine learning tools and techniques**, Segunda Edición, Morgan Kaufmann, 2005.

[11] J. Jang, "Structure determination in fuzzy modeling: A Fuzzy CART approach", **In proceedings IEEE Conference on Fuzzy Systems**, 1994, pp. 480–485,

[12] C.Z. Janikov, "Fuzzy Decision trees: Issues and Methods", **IEEE transactions on Systems, Man and Cybernetics**, Vol. 28, 1998, pp. 1–14.

[13] L. Koen-Myung, L. Jee-Hyong, L. Kyung-Mi, L. Hyung, "A Fuzzy Decision Tree Induction Method for Fuzzy Data", **In proceedings IEEE Conference on Fuzzy Systems**, 1999, pp. 22–25.

[14] R.E. Schapire, "Theoretical views of boosting", **Lecture Notes in Computer Science**, Vol. 1572, 1999, pp. 1–10.