Informe Técnico – Technical Report
DPTOIA-IT-2006-001
Marzo, 2006

Multiclasificadores: Métodos y Arquitecturas

Saddys Segrera Francia
María N. Moreno García

Departamento de Informática y Automática
Universidad de Salamanca

Revisado por:

Dr. Luis Alonso Romero

Dr. Ángel Luis Sánchez Lázaro

Aprobado en el Consejo de Departamento de 29 de Marzo de 2006

Información de los autores:

Saddys Segrera Francia

Estudiante de Doctorado

Departamento de Informática y Automática

Facultad de Ciencias – Universidad de Salamanca

Plaza de la Merced S/N – 37008 – Salamanca

saddyssf@hotmail.com

María N. Moreno García

Área de Lenguajes y Sistemas Informáticos

Departamento de Informática y Automática

Facultad de Ciencias – Universidad de Salamanca

Plaza de la Merced S/N – 37008 – Salamanca

mmg@usal.es

Resumen

Los multiclasificadores pertenecen a una reciente área de la minería de datos que ha permitido mejorar, en general, la precisión de las predicciones por medio de la combinación de clasificadores individuales. Un número creciente de trabajos de la investigación se dedica a la propuesta de nuevos algoritmos y métodos, que a su vez corrigen algunos de los problemas que presenta la combinación de clasificadores. El objetivo de este trabajo es mostrar la diversidad de aplicaciones de los multiclasificadores así como analizar los principales métodos y arquitecturas de construcción.

Abstract

Multiclassifiers belong to a recent area of data mining that has allowed improving, in general, the precision of the predictions by means of the combination of different classifiers. An increasing number of research works are dedicated to the proposal of new algorithms and methods, which in turn correct some of the problems that the combination of classifiers has. The aim of this paper is to show the diversity of multiclassifiers applications as well as to analyze the main methods and generation architectures.

Tabla de Contenidos

Tabla de Figuras

1 Introducción

Este trabajo presenta un estudio acerca de los multiclasificadores, que a diferencia de las técnicas de clasificación tradicional, están teniendo un amplio auge en el ámbito científico, debido a que tienen en cuenta todas las hipótesis válidas en mayor o menor grado con los datos, es decir, no se desecha ninguna de las consistentes con los datos y lo que se realiza es la combinación de las predicciones del conjunto. El uso de los multiclasificadores se ha incrementado debido a que resuelven los problemas de sobreadaptación (*overfitting*) y es posible obtener buenos resultados con pocos datos de entrenamiento. Asimismo, un problema complejo puede ser descompuesto en múltiples sub-problemas que sean más fáciles de entender y resolver, incluso los errores no-correlacionados de los clasificadores individuales pueden eliminarse por medio de la combinación [42], [68].

Los multiclasificadores son conjuntos de clasificadores diferentes que realizan predicciones que se fusionan y se obtiene como resultado la combinación de cada una de ellas. El hecho o la idea de combinación hace que los multiclasificadores sean citados en la literatura a través de distintos términos, entre ellos: métodos de ensamble [20], [85]; modelos múltiples *(multiple models*) [38], [55], [73]; sistemas de múltiples clasificadores (*multiple classifier systems*) [28]; combinación de clasificadores (*combining classifiers*) [42]; integración de clasificadores (*integration of classifiers*) [79]; mezcla de expertos (*mixture of experts*) [38]; comité de decisión (*decision committee*) [82]; comité de expertos (*committee of experts*); fusión de clasificadores (*classifier fusion*) [14], [53], [78] y aprendizaje multimodelo (*multimodel learning*) [34].

Se dividen en dos grupos: los métodos de ensamble o ensamblaje y los métodos híbridos, aunque existen otros métodos relacionados que también serán explicados. Los primeros se refieren a aquellos conjuntos de modelos que se combinan para crear un nuevo modelo, empleando para ello la misma técnica de aprendizaje. En el caso de los segundos, son combinaciones de algoritmos de aprendizaje que crean a su vez nuevas técnicas de aprendizaje híbridas.

Los multiclasificadores se distinguen unos de otros [47] atendiendo a diversas características. Entre ellas podemos citar: el número de clasificadores individuales acoplados, el tipo de cada clasificador (redes neuronales, árboles de decisión, vecino más cercano, etc.) [2], [3], [13], [32], [51], [63], [67], [69], [70], las características de los subconjuntos usados por cada clasificador del conjunto, la agregación de las decisiones particulares (voto mayoritario, asignación de pesos, subespacio de mejor comportamiento, funciones de promedio, máximo, mínimo, producto, etc.) [1], [9], [14], [37], [46], [53], [75], [86] y el tamaño y la naturaleza de los conjuntos de datos de entrenamiento para los clasificadores.

La evaluación y comparación entre los distintos multiclasificadores se establece a través de indicadores de rendimiento que incluyen el grado de generalización, aprendizaje, clasificación correcta e incorrecta y el tiempo real de ejecución [5].

Las aplicaciones en las que han sido empleados multiclasificadores son diversas. Existen muchos trabajos dirigidos a la construcción y aprendizaje de modelos predictivos dedicados al reconocimiento de la identidad a través de la verificación de distintas características biométricas [7], [10], [25], [39], [40], [44], [50], [57], [59], [78], en investigaciones bioinformáticas y médicas [21], [63], análisis geofísicos y teledetección [11], [29], entre otros muchos.

Este documento se ha organizado de la siguiente forma: La sección 2, explica las diferentes arquitecturas o topologías de los distintos métodos. La sección 3 introduce los tipos de mecanismos de combinación de los métodos de ensamble. La sección 4 describe los principales métodos de ensamble, la sección 5 expone los métodos híbridos más difundidos, la sección 6 analiza dos métodos no ensamble, relacionados con las aplicaciones de multiclasificadores, la

sección 7 muestra los resultados obtenidos en el caso de estudio. Finalmente, en la sección 8 se presentan las conclusiones.

2 Arquitecturas

Disponiendo de un conjunto de *L* clasificadores, la arquitectura depende de la forma en que se desea integrar al conjunto para garantizar una toma de decisión, ya sea independiente unos de otros, por eliminación de hipótesis (decisiones dependientes) o a través de la cooperación de clasificadores (cada uno soluciona un problema) [49], [62].

De acuerdo a ello los multiclasificadores pueden adoptar distintas arquitecturas: secuencial, en serie o vertical, paralela u horizontal e híbrida (mezcla de la arquitectura secuencial con la paralela, con interacción, etc.).

2.1 Arquitectura vertical o secuencial

El modelo en serie establece que la información resultante de un clasificador se convierte en información de entrada para otro. Un esquema de este modelo aparece en la figura 1.

Esta organización en niveles sucesivos de decisión permite reducir progresivamente el número de clases posibles. En cada nivel: un único clasificador tiene en cuenta la respuesta proporcionada por el clasificador colocado anteriormente para tratar los rechazos y confirmar la decisión obtenida en el eslabón anterior. Existen dos enfoques: la re-evaluación y la reducción del conjunto de clases [62].

En el primero el principio fundamental es asegurar la re-evaluación de los ejemplos que se rechazan o se reconocen con muy baja confianza en cualquier nivel.

Los datos de entrada se presentan al primer clasificador en la configuración. En lugar de indicar la clase a que los datos actuales podrían pertenecer, el primer experto genera un valor de confianza que se corresponde al indicador de decisión. Tres escenarios pueden ser considerados:

- La confianza es un valor superior al umbral establecido y la decisión es la aceptación. La re-evaluación en el siguiente nivel es innecesaria.
- La confianza es un valor que se encuentra dentro del rango definido por el primero y el segundo umbral, en este caso dependiendo de las restricciones se toma una decisión u otra.
- La confianza es un valor inferior al segundo umbral y la decisión es el rechazo, lo que hace necesaria la re-evaluación en el nivel posterior.

En las situaciones donde se necesita la re-evaluación, el clasificador siguiente realiza la búsqueda de una solución en todo dominio de clases; genera otro valor de confianza que indica la posible clase y este proceso se repite para los sucesivos clasificadores hasta que un clasificador encuentre un valor de confianza suficientemente alto o el clasificador final emite su decisión.

En la reducción del conjunto de clases el clasificador primario en la configuración genera una lista de posibilidades que constituye un subconjunto del número total de clases. El próximo clasificador limita su análisis al subconjunto de clases generado por el clasificador anterior y así sucesivamente.

La adopción de esta arquitectura está acompañada de un filtrado progresivo de las decisiones (reducción de la ambigüedad) y es sensible al orden en el cual se colocan los clasificadores, por lo que se debe tener un conocimiento *a priori* del comportamiento de cada uno de los clasificadores. De manera general, es difícil de optimizar el conjunto ya que existe dependencia.

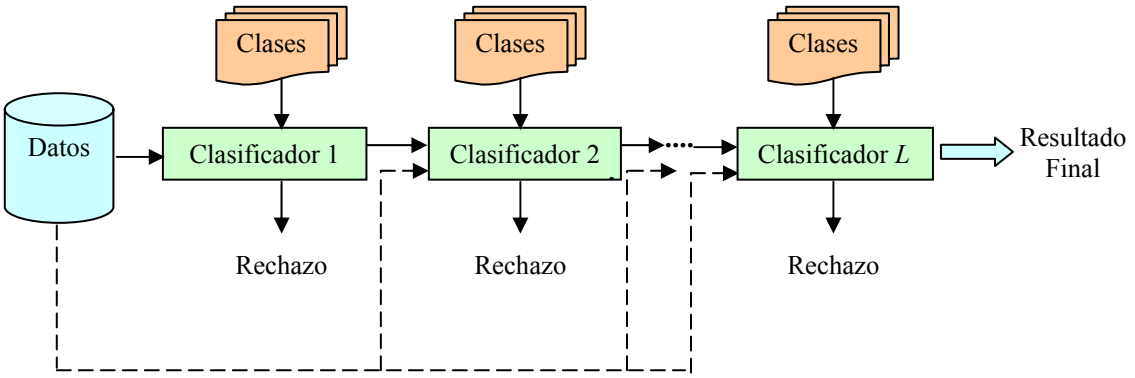

Figura 1. Arquitectura vertical o secuencial con el enfoque de re-evaluación.

2.2 Arquitectura horizontal o paralela

A diferencia de la arquitectura o topología secuencial, en la paralela los clasificadores operan independientemente unos de otros, luego se fusionan sus respectivas respuestas y se busca un consenso entre los clasificadores para llegar a una única decisión, véase la figura 2.

Este esquema es muy fácil de aplicar, no requiere de una reparametrización de los otros clasificadores en caso de que existan modificaciones en el conjunto. Su desventaja radica en que la activación de todos los clasificadores conlleva a un costoso tiempo de cálculo.

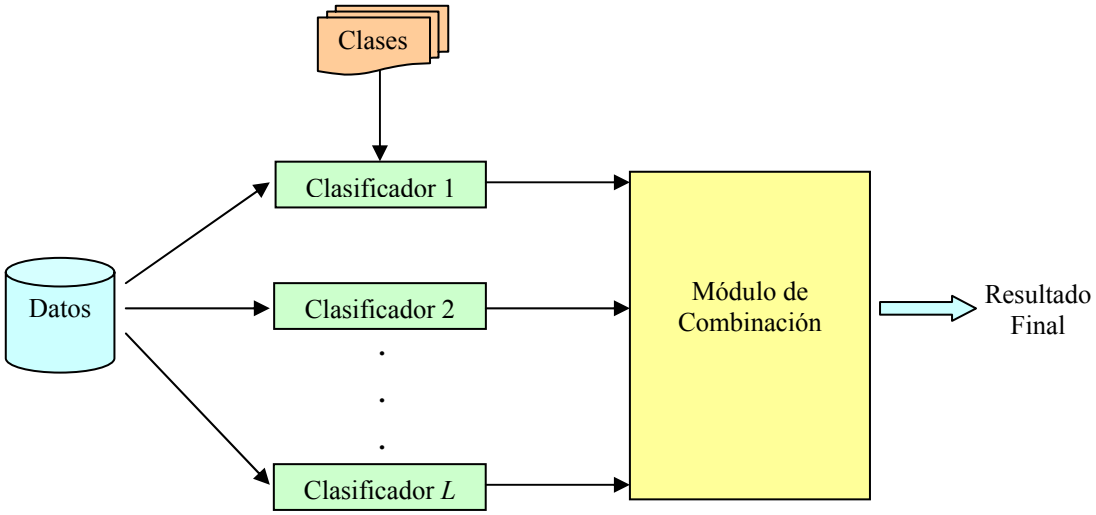

Figura 2. Arquitectura horizontal o paralela.

2.3 Arquitectura híbrida

En la arquitectura híbrida se combinan las ventajas de las arquitecturas secuencial y paralela, es decir, la reducción del conjunto de las clases posibles y la búsqueda de un consenso entre los clasificadores.

Al unir estas dos arquitecturas en una, se obtiene mejor provecho de cada uno de los clasificadores utilizados, existen ahora numerosos esquemas de combinación para extraer lo mejor de los datos. El esquema híbrido se muestra en la figura 3, donde los datos deben tratarse de forma dependiente y al igual que en la arquitectura secuencial, es aun más complejo de optimizar.

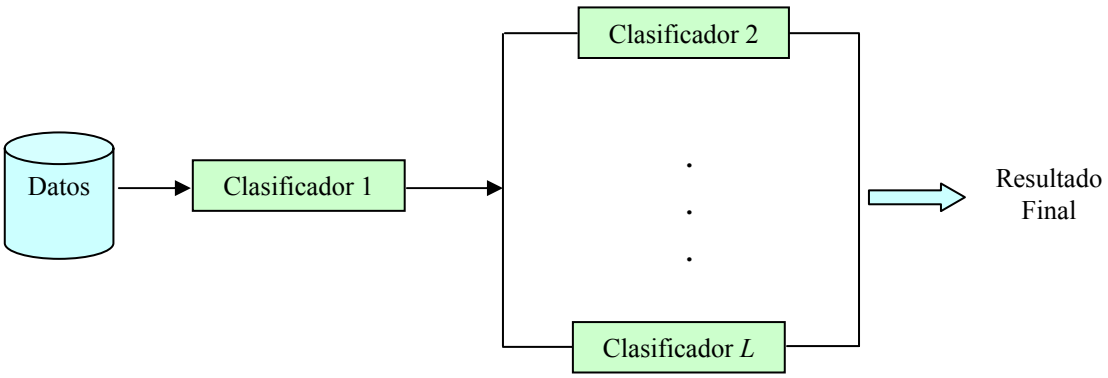

Figura 3. Arquitectura híbrida.

3 Estrategias de combinación

En esta sección se proporciona una descripción de cómo las salidas de cada clasificador se pueden fusionar en una sola para el conjunto. Aunque las salidas de los clasificadores individuales así como la del conjunto pueden tomar diferentes formas, la filosofía de la mayoría de las técnicas discutidas seguidamente se puede aplicar a cualquiera de ellos.

3.1 Métodos de nivel abstracto (*Abstract-level methods*)

En el nivel abstracto, la salida del clasificador es una etiqueta (también llamada salida *crisp*) asociada a la clase:

D: $\Re^n \rightarrow \Omega$

Donde D es una Función de decisión que asigna el vector de entrada de las características $x \in \Re^n$ en el espacio de la etiqueta $\Omega = \{c_1,...,c_k\}$. En la figura 4 se muestra un esquema donde la regla de combinación fusiona las salidas *crisp* de los clasificadores $S^{(i)} \in \Omega$ para N clasificadores y k clases.

Este método se subdivide en dos tipos de reglas. Las reglas fijas, basadas en voto mayoritario y las entrenadas, que incluyen las basadas en voto mayoritario con peso y las basadas en el enfoque de Bayes, respectivamente.

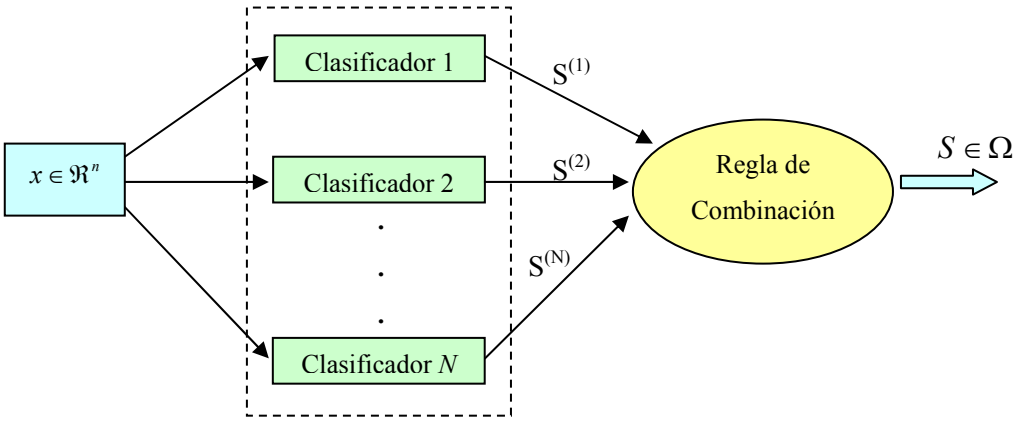

Figura 4. Esquema de un método de combinación de nivel abstracto.

3.1.1 Voto mayoritario simple

Esta técnica funciona de forma similar a cómo el ser humano vota en las elecciones políticas, es decir, cada clasificador tiene un voto que tiene igual valor. En esencia, la clasificación más popular es la que resulta elegida por el conjunto como decisión final. La variante más básica es el voto de la pluralidad, donde la clase con la mayoría de los votos gana [4].

La figura 5 muestra un ejemplo del esquema de este método para 3 clasificadores. Usualmente el número de clasificadores que se emplea es impar, la frecuencia de la clase ganadora debe ser al menos el 50%, es decir, *N/2*.

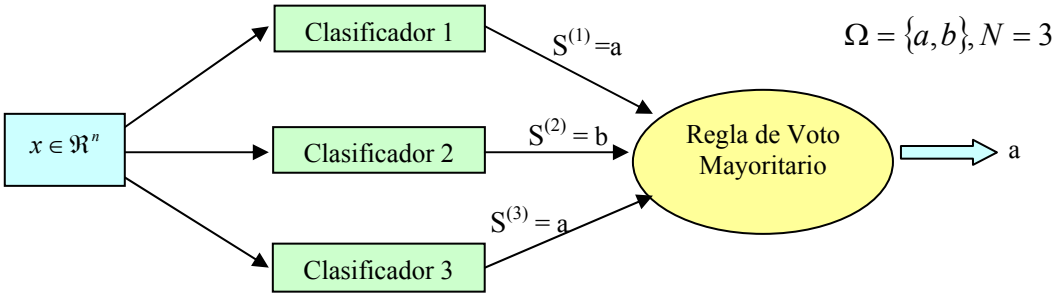

Figura 5. Ejemplo del método de voto mayoritario.

3.1.2 Voto mayoritario por peso

Esta técnica es similar a la manera en que los accionistas en las corporaciones emiten sus votaciones para llegar a una decisión final, esto significa que cada votante puede tener diferente grado de influencia en el resultado final [52]. Un enfoque de asignar pesos consiste en entrenar a cada clasificador individualmente, y hacer que cada clasificador evalúe de manera individual unos datos de prueba fijados. El peso asignado al voto de cada clasificador se basa entonces en su funcionamiento en el sistema de la prueba. Si se asume que los clasificadores son independientes y que los resultados obtenidos en la prueba son una representación exacta de la eficacia de cada clasificador, es probable que esta técnica trabaje mejor que la del voto mayoritario simple. La clase con mayor peso es la que se emite como resultado final del conjunto.

En la figura 6, partiendo del mismo ejemplo de tres clasificadores, se observa el esquema de aplicación de este método.

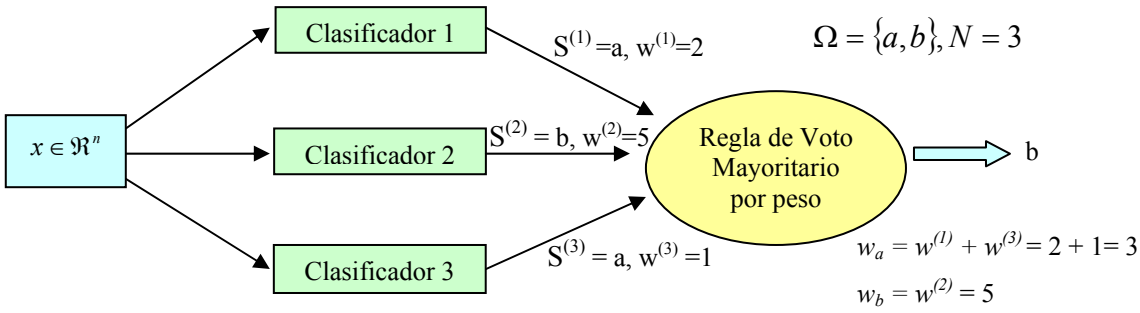

Figura 6. Ejemplo del método de voto mayoritario por peso.

Como se observa el peso de una clase se corresponde con la suma de los pesos de los clasificadores individuales donde fue predicha dicha clase. En el ejemplo la clase "a", fue predicha por el clasificador 1 y 3, por lo que el peso de la clase "a" es la suma de los pesos de estos dos clasificadores. La clase "b" sólo fue predicha por el clasificador 2, de ahí que el peso de esta clase es el peso de este clasificador. Nótese que para el mismo ejemplo mostrado en la figura 5, el resultado para el voto mayoritario simple no tiene que coincidir con el del voto mayoritario por peso, ya que en el ejemplo de la figura 6, aunque la clase "a" fue predicha por más clasificadores, la suma de sus pesos no supera a la del único clasificador que predijo a la clase "b".

3.1.3 Reglas basadas en el enfoque de Bayes

Este tipo de reglas de fusión es más complicado y menos popular para la combinación de las decisiones del conjunto de clasificadores. La estadística bayesiana se utiliza para determinar clasificaciones teóricamente óptimas del conjunto.

La entrada x es asignada a la clase c_i si su probabilidad posterior es:

$$P(c_i \left| S^{(1)}, S^{(2)}, ..., S^{(N)}) > P(c_j \right| S^{(1)}, S^{(2)}, ..., S^{(N)}), \forall j \neq i$$

Las reglas de fusión basadas en la Fórmula de Bayes intentan estimar, a través de un conjunto de entrenamiento/validación, estas probabilidades posteriores:

- Comportamiento del Espacio de Conocimiento - *Behavior- Knowledge Space* (BKS) [37].

- Regla bayesiana/Funciones de Credibilidad - *Belief Functions* [84].

Regla BKS

Las reglas BKS intentan estimar las probabilidades posteriores de un conjunto de entrenamiento/validación, a través del registro de la frecuencia de cada clase correspondiente a cada combinación de las decisiones de los clasificadores $S^{(1)},...,S^{(N)}$ [64].

$$\hat{P}(c_i \left| S^{(1)}, ..., S^{(N)}) = \frac{N(c_i \right| S^{(1)}, ..., S^{(N)})}{\sum_i N(c_i \left| S^{(1)}, ..., S^{(N)})}$$

Una vez estimadas estas probabilidades, las reglas BKS trabajan como una tabla que asocia la clasificación final a cada combinación de las salidas de los clasificadores, véase el ejemplo desarrollado por [66] en la figura 7, donde se emplean tres clasificadores $\{S(1), S(2), S(3)\}$ y dos clases con valores $\{0,1\}$.

En el ejemplo aparece una tabla que muestra la frecuencia de ocurrencia de las clases "0" y "1" para las 8 posibles combinaciones a partir de las salidas de los clasificadores. Se ha calculado la probabilidad de ocurrencia de la combinación $\{0,1,0\}$, en este caso se seleccionará como resultado final la clase "0" que posee mayor probabilidad (0.82) en comparación con la probabilidad de ocurrencia de la clase "1" (0.18).

El número total de probabilidades a calcular es de k^{N+1}, siendo k el número de clases y N el número de clasificadores, por lo que en el ejemplo serían 16. Si los valores de k y N son altos el número de probabilidades a estimar se incrementa rápidamente.

Clases	Resultados de las salidas de los clasificadores S(1), S(2), S(3)							
	0,0,0	0,0,1	**0,1,0**	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
0	100	50	**76**	89	54	78	87	5
1	8	88	**17**	95	20	90	95	100

$$P(c=0\left|S^{(1)}=0,S^{(2)}=1,S^{(3)}=0\right.)=\frac{76}{76+17}=0.82$$

$$P(c=1\left|S^{(1)}=0,S^{(2)}=1,S^{(3)}=0\right.)=\frac{17}{76+17}=0.18$$

Figura 7. Ejemplo de aplicación de las reglas BKS.

Regla Bayesiana

Esta regla también llamada *Belief Functions Rule* (BFR) o Regla de Funciones de Creencias se diferencia de la regla BKS, debido a que se asume que los clasificadores son independientes [66].

La fórmula de Bayes considera que:

$$P(c_i\left|S^{(1)},...,S^{(N)}\right.)\propto P(S^{(1)},...,S^{(N)}\left|c_i\right.)\cdot P(c_i)$$

Bajo la suposición de clasificadores independientes dada la clase, sería:

$$P(S^{(1)},...,S^{(N)}\left|c_i\right.)=\prod_j P(S^{(j)}\left|c_i\right.)\propto\prod_j P(c_i\left|S^{(j)}\right.)\cdot P(S^{(j)})$$

La estimación de $P(c_i\left|S^{(j)}\right.)$ se obtiene a través de una matriz de confusión de cada clasificador entrenado con un conjunto de entrenamiento/validación. Si $m_{ih}^{(j)}$ es el número de ejemplos entrenados con la verdadera c_i y es asignada la clase c_h para el *j-ésimo* clasificador:

$$P(clase=c_i\left|S^{(j)}=c_h\right.)=\frac{m_{ih}^{(j)}}{\sum_h m_{ih}^{(j)}}$$

Una ventaja de BFR en relación con BKS, es que reduce el número de parámetros, que en este caso es $N*k^2$. Otro elemento a favor, es que esta técnica trabaja bien incluso para conjuntos de entrenamiento pequeños. El principal problema de BFR es conocer cuando se puede asumir que existe independencia condicional entre los clasificadores.

3.2 Métodos de nivel de rango (*Rank level methods*)

Un clasificador asocia frecuentemente a cada clase un "valor", o una probabilidad, que indica su "grado de confianza" en la clasificación de un patrón *x* determinado. Esta información se puede utilizar para asignar un "rango" a cada clase.

Si $\Omega=\{c_1,...,c_k\}$ es un conjunto de clases, cada clasificador puede ofrecer una lista "ordenada" (por rango) de las clases. Las clases pueden ordenarse, por ejemplo, por el incremento de las probabilidades, es decir, el mayor rango estará asociado a la clase con mayor probabilidad. Si se tiene como valor la medida de "distancia" como un "grado de confianza", se puede asociar el máximo rango a la clase con la menor distancia.

Al igual que los métodos de nivel abstracto, los métodos de nivel de rango poseen reglas fijas y reglas entrenadas. En el primer caso está el método de la cuenta de Borda y en el segundo, el método de la cuenta de Borda por peso [15], [80].

3.2.1 Método de la cuenta de Borda

Sea N el número de clasificadores, k el número de clases, se denotará como $r_i^{(j)}$ al rango del clasificador *j-ésimo* asociado a la clase i. De manera que:

$r_i = \sum_{j=1}^{N} r_i^{(j)}$ será el rango total asociado a la clase i. La clase ganadora será aquella con un valor de rango total mayor. Un ejemplo de la aplicación de este método se muestra en la figura 8, donde se han utilizado 3 clasificadores y existen 4 clases *{a,b,c,d}*. En el ejemplo el mayor valor total de los $r_i^{(j)}$ fue para la clase *b* con un valor de 10.

Este método no requiere de entrenamiento y es muy simple de procesar, además no tiene en cuenta la técnica empleada por los clasificadores individuales, no posee una base teórica, pero depende grandemente de la definición de los rangos.

Valor del rango	Clasificador 1	Clasificador 2	Clasificador 3
4	c	a	b
3	b	b	a
2	d	d	c
1	a	c	d

$r_a = r_a^{(1)} + r_a^{(2)} + r_a^{(3)} = 1 + 4 + 3 = 8$

$r_b = r_b^{(1)} + r_b^{(2)} + r_b^{(3)} = 3 + 3 + 4 = 10$

$r_c = r_c^{(1)} + r_c^{(2)} + r_c^{(3)} = 4 + 1 + 2 = 7$

$r_d = r_d^{(1)} + r_d^{(2)} + r_d^{(3)} = 2 + 2 + 1 = 5$

Figura 8. Ejemplo de aplicación del método de la cuenta de Borda.

3.2.2 Método de la cuenta de Borda por peso

En el caso de este método y teniendo en consideración la técnica empleada por los clasificadores individuales, se puede asignar pesos a los rangos, es decir, los pesos están asociados a los clasificadores:

$$r_i = \sum_{j=1}^{N} w_j r_i^{(j)}$$

Estos pesos pueden estar basados en la precisión de cada clasificador, medida en su funcionamiento con los datos de entrenamiento.

La distribución de los rangos producidos por los clasificadores se puede utilizar para denotar la calidad del patrón de entrada: por ejemplo, un alto grado de acierto puede indicar un patrón fácilmente reconocible. Los valores de los rangos se pueden utilizar en la combinación de clasificadores con arquitecturas en serie, con el objetivo de reducir el número de las clases para la clasificación posterior.

Sin embargo, si las mediciones se producen para la salida, puede ser conveniente utilizar estas medidas en vez de los valores de los rangos, ya que estas mediciones constituyen una fuente más rica de información.

3.3 Métodos de nivel de medidas

Algunos clasificadores pueden ofrecer como salida valores de confianza o medidas de distancia para cada clase $\omega_1,...,\omega_k$, para un patrón de entrada *x*. Estas medidas pueden ser interpretadas como la probabilidad de que un patrón pertenezca a las diferentes clases.

Las reglas de combinación en el nivel de medidas se basan en computar una nueva medida de confianza para cada clase. Generalmente, las salidas de los clasificadores se normalizan en el rango [*0;1*] antes de efectuarse la combinación. La normalización de las salidas de los clasificadores no es una tarea trivial cuando los clasificadores se combinan con diferentes rangos de salida y diferentes tipos de salida.

Existen también, en este tipo de método, para alcanzar la combinación de clasificadores, reglas fijas (promedio, producto, máximo, mínimo, mediana y otros operadores estadísticos) y reglas entrenadas (promedio pesado, enfoque "*stacked*").

3.3.1 Reglas de promedio simple, el producto y otros operadores estadísticos

Regla del promedio simple

Una de las reglas de combinación más simples se basa en promediar las salidas de los clasificadores individuales.

Dado un problema con *k* clases, se define el vector *v(k)* de *k* componentes (clases) como vector de probabilidades, la suma de los componentes debe ser igual a 1. Con *m* clasificadores, se unifican en único vector α. Este vector se utiliza para extraer la clase predicha, que será la clase que obtenga la mayor probabilidad [33]. Lo que significa que existen tantos vectores como clasificadores.

De modo que el promedio se calcula como la suma de las probabilidades de una misma clase en todos los clasificadores dividido entre el número de éstos:

$$\alpha = \sum_{j=1}^{m} \frac{v_j}{m}$$

Es bien conocido que promediando se puede reducir la varianza de los errores de estimación.

Regla del producto

Si los vectores de las características usados por los clasificadores individuales son distintos, se puede utilizar la regla del producto.

Teniendo en cuenta lo explicado en el epígrafe anterior se puede establecer la regla del producto como la multiplicación de las probabilidades de una misma clase en todos los clasificadores. Que sería:

$$\alpha = \prod_{j=1}^{m} v_j$$

Otros operadores estadísticos

Otros operadores estadísticos son la suma, la mediana, el máximo y el mínimo. El primero de estos operadores se calcula sumando las probabilidades de una misma clase en todos los clasificadores.

La mediana se calcula ordenando de menor a mayor los valores de las probabilidades de una misma clase en todos los clasificadores. Si el número de valores es impar, la mediana es el

valor del individuo que ocupa el valor central y si el número de términos es par, la mediana es el valor medio de los datos centrales.

El máximo busca el valor superior de las probabilidades de una misma clase en todos los clasificadores. En el caso del mínimo se aplica la misma regla pero se busca el valor inferior. Que sería:

Suma: $\alpha = \sum_{j=1}^{m} v_j$

Mediana: $\alpha = mediana_{i \le j < m}(v_j)$

Máximo: $\alpha = máximo_{i \le j < m}(v_j)$

Mínimo: $\alpha = mínimo_{i \le j < m}(v_j)$

En todos los casos, una vez calculados los valores por cada clase, la clase que es predicha por el conjunto se corresponde con la que posea el mayor valor calculado para cualquiera de estos operadores.

En [45] se concluyó que para problemas de dos clases, las funciones de máximo y mínimo son las mejores estrategias, seguidas del promedio.

Las reglas del producto y del mínimo se derivan bajo la hipótesis de que los clasificadores son estadísticamente independientes.

Las reglas de la suma, el máximo y la mediana se derivan bajo la hipótesis de que las probabilidades estimadas a posteriori por los clasificadores no se desvíen significativamente de la clase de probabilidades anteriores.

La aplicación de los operadores estadísticos antes mencionados se ilustra en el siguiente ejemplo. Si se tienen tres clasificadores individuales (*m=3*) para resolver un problema de 4 clases (*k=4*) con los valores {*a, b, c, d*} y las predicciones de los clasificadores para cada clase son:

v_1={*0.25, 0.55, 0.10, 0.10*}

v_2={*0.50, 0.00, 0.05, 0.45*}

v_3={*0.15, 0.20, 0.30, 0.45*}

Entonces los resultados serían:

Suma: $\{0.90, 0.75, 0.45, 1.00\} \rightarrow d$

Promedio: $\{0.30, 0.25, 0.15, 0.33\} \rightarrow d$

Producto: $\{0.018, 0.00, 0.0015, 0.020\} \rightarrow d$

Mediana: $\{0.25, 0.20, 0.10, 0.45\} \rightarrow d$

Máximo: $\{0.50, 0.55, 0.30, 0.45\} \rightarrow b$

Mínimo: $\{0.15, 0.00, 0.05, 0.10\} \rightarrow a$

3.3.2 Operadores pesados

Un aspecto de reglas fijas es la introducción de pesos a las salidas de los clasificadores. Un simple criterio es introducir los pesos proporcionalmente a la exactitud de cada clasificador

individual. Los pesos también pueden ser calculados extrayéndolos de la matriz de confusión de cada clasificador.

Entre estas reglas están el promedio y el producto pesado. Sea w_j el peso asociado al clasificador *j-ésimo*.

Los pesos son usados como factores de las salidas al calcular el promedio ponderado:

$$p_i = \sum_{j=1}^{N} w_j p_i^j$$

También los pesos pueden ser usados como exponentes de las salidas a través del producto pesado:

$$p_i = \prod \left(p_i^j\right)^{w_j}$$

3.4 Comparación entre los métodos de fusión

Un resumen comparativo entre los 3 grandes tipos de métodos de fusión aparece reflejado en el Cuadro 1.

Cuadro 1. Comparación entre los diferentes grupos de métodos de fusión

Características	Métodos de fusión		
	Nivel abstracto	**Nivel de rango**	**Nivel de medidas**
Ventajas	• Pueden ser aplicados para cualquier fusión de clasificadores.	• Es conveniente en problemas con muchas clases, donde la clase correcta puede aparecer a menudo cerca de la lista superior. • Puede ser preferido a las salidas para evitar la carencia de la consistencia al usar diversos clasificadores. • Puede ser preferido a las salidas para simplificar el diseño de combinación.	• Puede explotar una cantidad de información más alta con respecto a los otros métodos de fusión. • Los combinadores complejos pueden ser diseñados y requeridos con frecuencia por clasificadores que exhiben diferentes funcionamientos y correlaciones complejas.
Desventajas	• Las reglas entrenadas imponen altas demandas en la calidad y el tamaño del conjunto de datos.	• No están soportados sobre una base teórica. • Los resultados dependen en la escala de los números asignados a las diferentes opciones.	• Se requiere normalización de las salidas de los clasificadores si se utilizan diferentes clasificadores. • Es necesario con frecuencia conjuntos de datos de gran tamaño y buena calidad.

Como se ha descrito en epígrafes anteriores, en los distintos grupos de métodos de fusión existen reglas fijas y entrenadas.

De manera general, las reglas fijas son simples, los requerimientos de cómputo son insignificantes y se adaptan bien para los conjuntos de clasificadores con independencia (pequeños errores correlacionados y con funcionamientos similares). Por su parte, las reglas entrenadas se caracterizan por su alto costo computacional, son más flexibles en comparación con las reglas fijas y son más idóneas que las reglas fijas para los clasificadores correlacionados o que exhiben funcionamientos diferentes.

4 Métodos de ensamble

4.1 Características de los métodos de ensamble

Como ya se ha explicado, de forma general, la combinación de clasificadores muestra mayor precisión que cualquiera de ellos de manera individual.

Una condición necesaria y suficiente para que se cumpla lo antes señalado es que los clasificadores individuales sean precisos y diversos.

Un clasificador se considera preciso si su error es inferior a 0.5. Por otro lado, dos clasificadores individuales son diversos cuando sus errores de salida no son correlacionados.

En [48] se estudiaron 10 medidas de diversidad para métodos de ensamble (estadísticas Q, coeficiente de la correlación, medida de discordancia, medida de doble fallo, medida de entropía, varianza de Kohavi-Wolpert, medida del acuerdo entre las evaluaciones de la capacidad de los asesores independientes (*interrater agreement*), medida de "dificultad", diversidad generalizada y diversidad de fracaso coincidente).

Estas medidas no deben ser un reemplazamiento para la estimación de la exactitud del conjunto sino que deben provenir del concepto intuitivo de la diversidad. Por ejemplo, la medida de diversidad de doble fallo (DF) se relaciona claramente con la exactitud del conjunto, es decir, los valores más pequeños de la diversidad medidos con este indicador (la mayor diversidad) favorecerá a los conjuntos más exactos.

Las razones que hacen superior el uso métodos de ensamble en lugar de un simple clasificador han sido explicadas de forma general en [18] y representadas en la figura 9:

- **Problema estadístico**: se presenta cuando el algoritmo que aprende está buscando un espacio de las hipótesis *H* que es demasiado grande para la cantidad de datos de entrenamiento disponibles. En tales casos, pueden existir diversas hipótesis que arrojan la misma exactitud en los datos del entrenamiento, y el algoritmo que aprende debe elegir una de éstas para la salida. Existe el riesgo de que la hipótesis elegida no prediga de forma correcta a los futuros datos. Un voto simple de todos estos clasificadores, que son igual de buenos puede reducir este riesgo.
- **Problema computacional**: en los casos donde existen suficientes datos de entrenamiento (por lo cual no se está en presencia el problema estadístico), el problema puede ser aun difícil de resolver computacionalmente por el algoritmo de aprendizaje para encontrar la mejor hipótesis. Un ensamble construido mediante la búsqueda local a partir de diferentes puntos de partida puede proporcionar una mejor aproximación a la verdadera función desconocida que cualquiera de los clasificadores individuales.
- **Problema de representación**: se presenta cuando el espacio de hipótesis no contiene ninguna hipótesis que sea una buena aproximación a la función verdadera *f*, que aprende

el algoritmo. En algunos casos, una suma de pesos de las hipótesis amplía el espacio de las funciones que pueden ser representadas. Por lo tanto, tomando un voto ponderado de las hipótesis, el algoritmo de aprendizaje puede ser capaz de formar una aproximación más exacta a *f*.

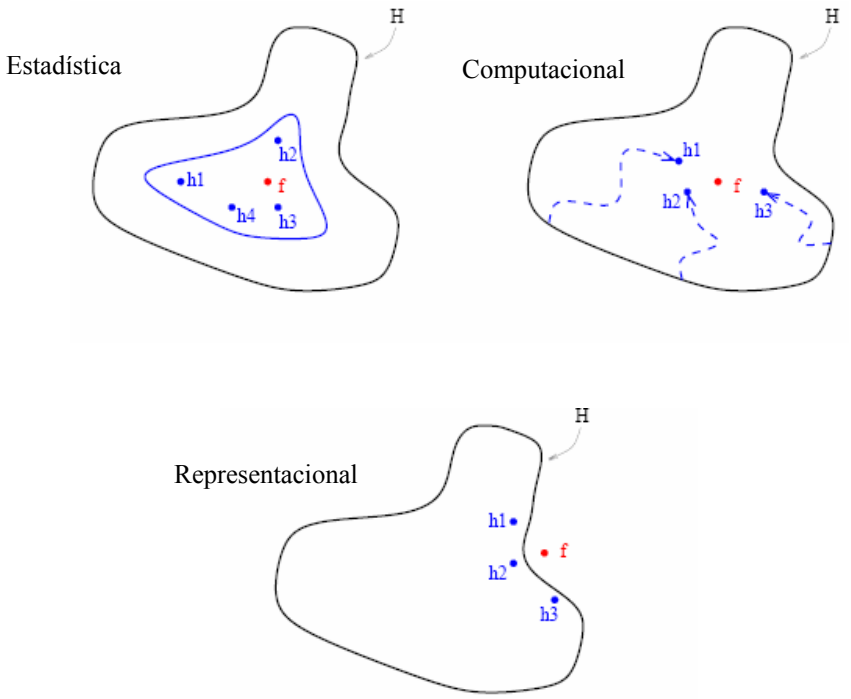

Figura 9. Tres razones fundamentales que hacen superior a un método ensamble de un clasificador único, tomado de [18].

4.2 Bagging

Bagging es un método propuesto por [8] y está basado en los conceptos de *bootstrapping* y agregación, de esta forma se reincorporan los beneficios de ambos y se le da nombre al método (Bootstrap AGGregatING). *Boostrapping* se basa en la generación de conjuntos de entrenamientos aleatorios con reemplazamiento. Una muestra *bootstrap* se genera al muestrear uniformemente instancias del conjunto de entrenamiento de manera aleatoria. De modo que se crean tantas muestras *bootstrap* como clasificadores existan, de ahí que cada clasificador se entrena con una réplica *bootstrap*, véase la figura 10. El método consiste en que los clasificadores individuales acoplados durante *K* iteraciones extraigan cada vez una muestra aleatoria del subconjunto *É* de *n* ejemplos con reemplazamiento a partir del conjunto original de ejemplos (pueden haber ejemplos repetidos); el subconjunto extraído aprende un modelo (técnica de aprendizaje a partir de una evidencia). Para clasificar un ejemplo *e*, se predice la clase de ese ejemplo para cada clasificador y se selecciona la clase con mayor voto [33], [71]. Es decir, Bagging utiliza para emitir la decisión final, el método del voto mayoritario, descrito en un epígrafe anterior.

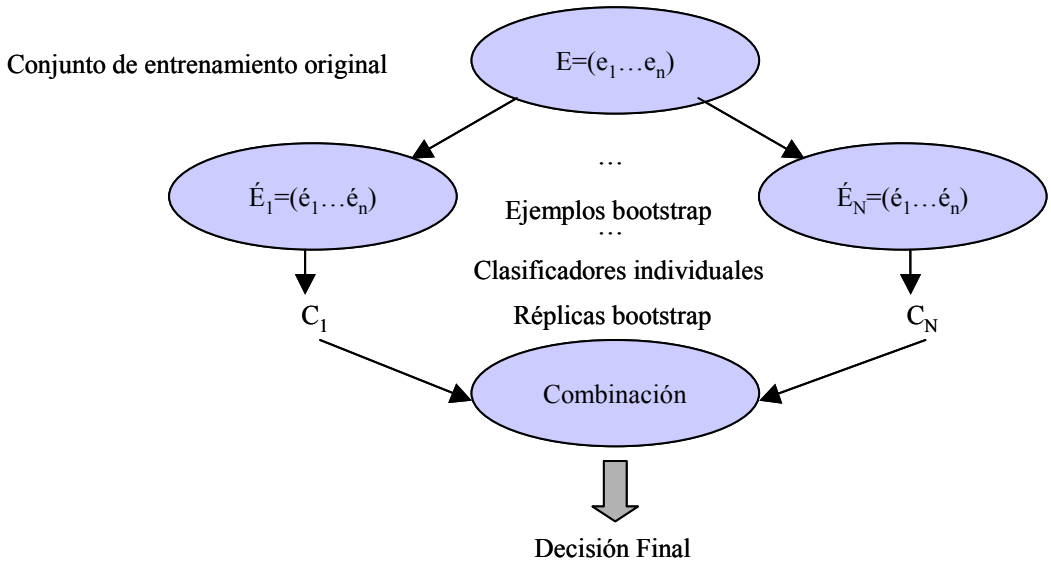

Figura 10. Esquema del método Bagging.

4.3 Boosting

Este método fue propuesto por [26]. El mecanismo se basa en la asignación de un peso a cada ejemplo del conjunto de entrenamiento. En cada iteración, este método aprende un modelo que minimiza la suma de los pesos de los ejemplos clasificados erróneamente. Además, los errores de 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 errados y se reduce el peso de los ejemplos acertados [33].

AdaBoost es la variante de Boosting más conocida. En un ciclo se aprende un modelo a través de la evidencia ponderada, se estima el error del modelo y dependiendo del valor del error se detiene el algoritmo o se continua el proceso repitiendo el ciclo, de ser así se actualiza los pesos de los ejemplos clasificados de forma acertada, se almacena el modelo y se efectúa la normalización del peso de todos los ejemplos (véase el algoritmo en la figura 11). A diferencia del método Bagging, este método considera como criterio de parada el valor del error. Es decir, mientras que en Bagging se predetermina el número de iteraciones para concluir el método, Boosting considera que si el error es igual o superior a 0.5 o no se puede disminuir (igual a 0), se detiene el algoritmo. Además, garantiza que se incrementen los pesos de los ejemplos mal clasificados para que en la siguiente iteración tengan mayor preferencia.

```
ALGORITMO Boosting(k:iteraciones,E:conjunto de ejemplos,A:Algoritmo de aprendizaje)
    i ← 1
    C ← Ø
    PARA CADA ejemplo e de E
    e_p ← p                          //Asigna a todos los ejemplos de E el peso p
    FIN PARA
    REPITE
        m ← A(É)                     //Aprende un modelo con A utilizando el conjunto con pesos E
        ε ← error de m sobre E
        SI (ε = 0) O (ε ≥0,5) ENTONCES termina bucle
        FIN SI
        PARA CADA ejemplo e de E
          SI la clase de e se ha predicho correctamente
          ENTONCES e_p ← e_p * ε/(1 - ε)
          FIN SI
        FIN PARA
        Normaliza(E)                 // Normaliza los pesos de los ejemplos
        {M} ← {M} ∪ m
    HASTA i = K
FIN ALGORITMO

Para clasificar un ejemplo e:
        PARA CADA clase c
        c_p ← 0                                  // asigna peso 0 a todas las clases
        FIN PARA
        PARA CADA modelo m de M
          t ← m(e)                               // t es la clase predicha por m
          c_t ← c_t – log(ε/(1 - ε))             // actualiza peso de la clase t
        FIN PARA
        DEVUELVE la clase con mayor c_p          // peso
```

Figura 11. Algoritmo Boosting, tomado de [33].

4.4 Método de Subespacios Aleatorios (*Random Subspace Method*)

El método de subespacios aleatorios (*Random Subspace Method*) o RSM [35] consiste en seleccionar aleatoriamente un cierto número de subespacios a partir del espacio original de las características, y utilizar un clasificador para cada subespacio (véase la figura 12). Es decir, cada clasificador trabaja con una parte del conjunto de atributos de los datos originales.

Sea n el número de ejemplos en el conjunto de entrenamiento y p el número de características o atributos, de modo que cada ejemplo del espacio original de entrenamiento puede ser representado por el vector $X_i = (x_{i1}, x_{i2}, ..., x_{ip})$ $(i=1,..., n)$. En el RSM se seleccionan los subespacios de las características aleatoriamente donde el número de atributos por cada subespacio es r, siendo $r < p$. Cada clasificador se entrena con un subespacio diferente *r-dimensional*. El método funciona de la forma:

1. Este paso se repite para $b = 1, 2, ..., B$, donde B es el número de clasificadores:
   a) Se selecciona un subespacio aleatorio *r-dimensional* $\widetilde{X}_b$ a partir del espacio original de las características *p-dimensional X*.
   b) Se entrena el clasificador C_b con $\widetilde{X}_b$.

2. Se combinan las decisiones de los clasificadores por voto mayoritario.

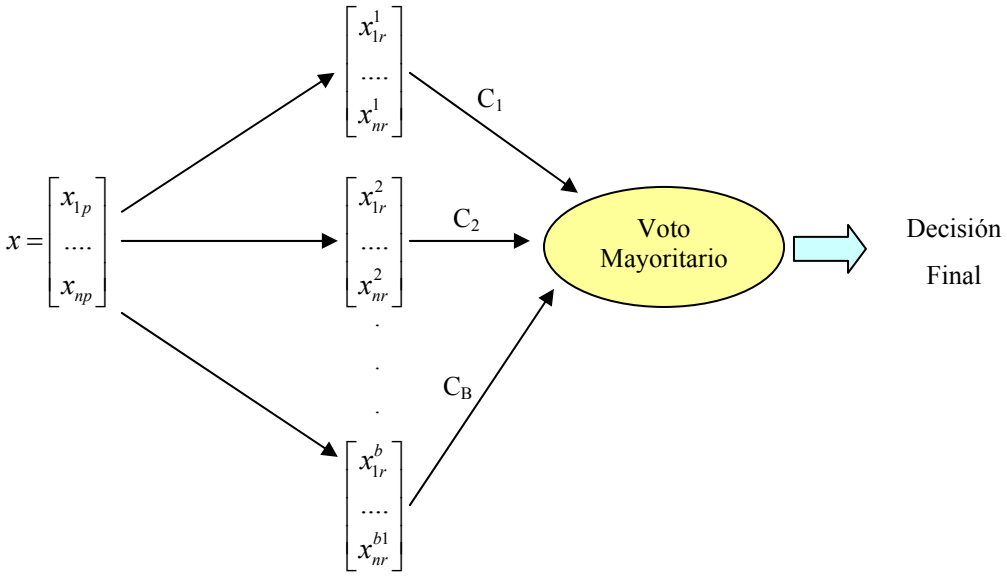

Figura 12. Esquema del método RSM.

En este método cada clasificador es un árbol de decisión y el resultado de la combinación es un árbol del tipo Forest. Las decisiones de n_t árboles individuales se combinan promediando la probabilidad condicional de cada clase en las hojas. Para un ejemplo x, $v_j(x)$ es el nodo terminal cuando se desciende en el árbol T_j (j=1, 2, ..., n_t). Dado esto, la probabilidad de x de pertenecer a la clase c (c=1, 2, ..., n_c) será denotada por $P(c|v_j(x))$:

$$P(c|v_j(x)) = \frac{P(c|v_j(x))}{\sum_{k=1}^{n_c} P(c_k, v_j(x))}$$

La probabilidad viene dada por la fracción de los ejemplos de la clase c sobre el total de los ejemplos asignados al nodo $v_j(x)$ (en el conjunto de entrenamiento). Un elemento importante tratado con frecuencia en la literatura es el criterio para la construcción del árbol. Ciertos tipos de divisiones pueden ser producidos hasta que el espacio de las características se reparte en las regiones que contienen solamente muestras de una sola clase (dado que no hay ambigüedades intrínsecas entre clases). En este contexto, la mayoría de los nodos terminales contienen solamente una clase y así el valor de la estimación $\hat{P}(c|v_j(x))$ es casi siempre 1. La función discriminante se define como:

$$g_c(x) = \frac{1}{n_t} \sum_{j=1}^{n_t} \hat{P}(c|v_j(x))$$

y la regla de la decisión es asignar x a la clase c para la cual el $g_c(x)$ es el máximo. Para los que contienen muestras de una clase, la decisión obtenida usando esta regla es equivalente a un voto de la pluralidad entre las clases decididas por cada árbol.

RSM se adapta especialmente bien a los espacios de características de grandes dimensiones que contienen características redundantes, pues no sufre del problema de la dimensionalidad. Los parámetros críticos para los resultados son: el número de subespacios aleatorios (el número de clasificadores) y la dimensionalidad de los subespacios aleatorios.

4.5 Codificación de la Salida de Corrección del Error (*Error-Correcting Output Coding*)

En [19] se describió la técnica *Error-Correcting Output Coding* (ECOC), la cual utiliza la manipulación de los valores de salida que entrega el algoritmo de aprendizaje.

Un *error-correcting code* (código de corrección del error) es una matriz de valores binarios como se muestra en la figura 13. La longitud del código es el número de columnas, el número de filas en el código se corresponde con el número de clases del problema a resolver. Un *codeword* es una fila en el código.

ECOC es muy utilizado en el reconocimiento de patrones, ya que trata de identificar los valores de entrada a partir de la salida.

Por ejemplo, en la identificación de dígitos manuscritos se tienen 10 clases posibles $K=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, la tarea de aprendizaje se descompone en m clasificadores, cada uno se especializa en resolver uno de los m sub-problemas.

A partir de K clases se construyen los *codewords* de m bits. El enfoque más simple es seleccionar los *codewords* aleatoriamente.

Los 4 métodos para construir buenos códigos de salida con esta técnica son [19]:

- Una técnica exhaustiva
- Un método que seleccione las columnas a partir de un código exhaustivo
- Un método basado en un algoritmo de *hill-climbing* (subida de la colina) aleatorizado
- Códigos BCH: abreviación de código Bose-Chaudhuri-Hochquenghem [6], [36]

De ahí que se construyen clasificadores binarios que aprenden funciones para emitir un resultado. Posteriormente los resultados de los clasificadores se combinan por el voto mayoritario o por la mínima distancia de Hamming.

En el caso del voto mayoritario por cada coincidencia del *codeword* generado con los de cada uno en la tabla por clasificador se recibe un voto. Después de que cada uno de los L clasificadores haya votado, la clase con el número mayor de votos se selecciona como la predicción del conjunto de clasificadores.

Si se selecciona la segunda forma de fusión para clasificar cada manuscrito todos los clasificadores trabajan y predicen la clase cuyo *codeword* esté más cercano a la distancia de Hamming.

Siguiendo el ejemplo mostrado en la figura 13, si se desea clasificar un nuevo dígito manuscrito x, las 6 funciones f_{vl}, f_{hl}, f_{dl}, f_{cc}, f_{ol} y f_{or} se evalúan para obtener un *codeword*, por ejemplo *110001*. Entonces se calcula la distancia para las 10 filas (clases). El *codeword* más cercano según la distancia de Hamming (se cuenta el número de dígitos que difieren), sería *110000*, el cual se corresponde a la clase "4", entonces se predice que $f(x)=4$.

Este método es idóneo para un número grande de K clases. Sin embargo, para generar un buen ECOC se deben satisfacer dos propiedades: la separación de las columnas y la separación de las filas.

La separación de las filas se refiere a la distancia entre cada *codeword* y los otros *codewords*, que define la distancia de la clase en cuestión a las otras clases. En cuanto a la separación de las columnas se entiende a que cada función f_i debe no correlacionarse con cada función f_j, es decir, la distancia entre cada columna debe ser suficientemente grande (en el sentido de la distancia de

Hamming). Es por ello que se debe evitar las correlaciones positivas o negativas entre columnas [34].

Los resultados no dependen del tamaño de los datos de aprendizaje, ni de la asignación particular de los *codewords* a las clases. ECOC proporciona una medida de confianza de clasificación que es tan buena como los otros métodos [19].

Clase	*Codeword*					
	lv	lh	ld	cc	ci	cd
0	0	0	0	1	0	0
1	1	0	0	0	0	0
2	0	1	1	0	1	0
3	0	0	0	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	0	0	1	0	0
9	0	0	1	1	0	0

Abreviatura	Significado
lv	Contiene línea vertical
lh	Contiene línea horizontal
ld	Contiene línea diagonal
cc	Contiene curva cerrada
ci	Contiene curva izquierda abierta
cd	Contiene curva derecha abierta

Figura 13. Ejemplo de *Error-Correcting Output Coding*, tomado de [19].

4.6 Comités de validación cruzada (*Cross-Validated Committees*)

La idea es dividir los datos de entrenamiento en varios conjuntos. Los conjuntos no tienen que ser necesariamente mutuamente exclusivos, ellos pueden compartir parte del conjunto (solaparse). Esta idea es similar a los métodos de remuestreo como la validación cruzada (*cross-validation*) propuesto en [74] y detallado en [81], que es utilizado en estadística para estimar el error de un predictor a partir de los datos disponibles. En el contexto de "comité", esta técnica se reforma para construir conjuntos de entrenamiento diferentes del conjunto original, de ahí el nombre de comités de validación cruzada (CVC) [60].

El algoritmo es bastante similar al procedimiento usado en la estimación del error de la predicción. Primero, se generan réplicas a partir del conjunto de entrenamiento original mediante la exclusión de un fragmento de los datos. Se denota D como los datos originales y D^{-v} denota los datos con el subconjunto v excluido. El procedimiento consiste en ir rotando para que cada elemento esté por lo menos una vez como fragmento excluido. Se generan réplicas D_1^{-v1},..., D_K^{-vk} y se entrena a cada clasificador (una red, en este caso) que forma parte del comité, véase la figura 14.

Los estimadores de error se basan en calcular la proporción de las instancias incorrectamente etiquetadas por el clasificador. Una vez construido el clasificador d y dado una instancia (X_i, c_i):

$$\Delta(X_i, c_i) = \begin{cases} 1, & si \;\; d(X_i) \neq c_i \quad (error) \\ 0, & si \;\; d(X_i) = c_i \quad (acierto) \end{cases}$$

Para todo k, $k = 1, 2, ...,K$, se construye un clasificador d_k, usando $D - D_k$ como conjunto de aprendizaje. Como ninguna de las instancias de D_k se ha usado para construir d_k, el estimador mediante el conjunto de prueba de d_k es $R(d_k)$:

$$R(d_k) = \frac{1}{|D_k|} \sum_{(X_i,c_i)\in D_k} \Delta(X_i, c_i)$$

Donde Δ se evalúa sobre d_k y D_k es el conjunto de prueba. Al finalizar se obtiene K clasificadores, d_k, con sus correspondientes estimaciones de error $R(d_k)$.

Posteriormente, se usa el mismo procedimiento para construir el clasificador d con todas las instancias de D.

Como cada d_k se construye con $D–D_k$, si K es grande, $|D - D_K| = N\left(1 - \frac{1}{K}\right) \approx |T|$. Este procedimiento es estable (todos los clasificadores d_k tienen una tasa de error aproximadamente igual a la del clasificador d).

Cuando $K = N$, el estimador por validación cruzada con N (tamaño del conjunto de aprendizaje)conjuntos se conoce como el estimador "que deja uno fuera" (del inglés *leave-one-out*). Para cada n, $n = 1, 2,..., N$ el *n-ésimo* ejemplo es descartado y el clasificador se construye utilizando los restantes $N - 1$ ejemplos. Entonces, el ejemplo descartado se usa para prueba y se estima el error mediante la ecuación:

$$R^{cv}(d) = \frac{1}{K} \sum_{k=1}^{K} R(d_k)$$

El algoritmo:

1. Dividir los datos en *v-fragmentos* $d_1,..., d_v$
2. *Leave-one-out* (dejar un fragmento fuera) d_k y entrenar la red h_k con el resto de los datos $(D - d_k)$
3. Usar d_k como un criterio de parada de la validación, si es necesario
4. Construir un comité (unir los resultados de los clasificadores) a partir de las redes usando un procedimiento de promedio simple.

Este algoritmo requiere de un gran esfuerzo computacional, ya que todos los ejemplos de D se usan para construir los d_k, y cada uno de ellos se usa exactamente una vez para prueba. No obstante, es adecuado para subconjuntos de pequeño tamaño.

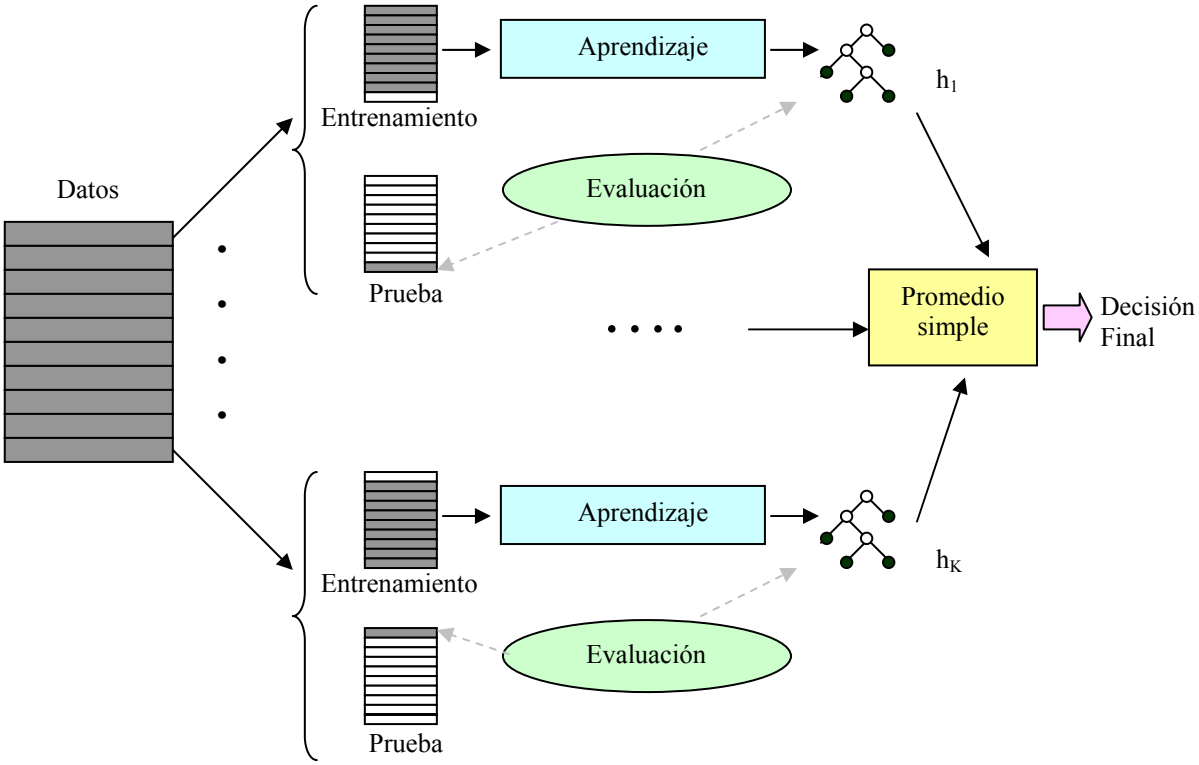

Figura 14. Algoritmo de Comités de Validación Cruzada, basado en [33].

4.7 Aleatorización (*Randomization*)

Esta modalidad de métodos de ensamble, se basa en incorporar aleatoriedad dentro de una técnica de aprendizaje estándar [15].

En [43] los autores exploraron el efecto de la selección del peso inicial en las redes neuronales que aprenden funciones simples con la técnica de retropropagación. Los resultados de sus experimentos demostraron la sensibilidad extrema de la retro-propagación (*back propagation*) al configurar los pesos iniciales.

Por otra parte, en las investigaciones presentadas en [65], también con redes, se probó que el nivel del ruido afecta a la independencia entre los conjuntos de entrenamiento, y de esta forma se logra una mejora relativa del conjunto. De ahí que, los autores realizaron estudios para inyectar ruido a los datos. El nivel del ruido afectó la calidad de cada clasificador por separado, aumentando su variación en el incremento de la variabilidad de los datos. Esto significa, que debe existir un nivel óptimo de ruido que conduce al funcionamiento óptimo del conjunto, que no tiene que coincidir con el real.

Otro ejemplo de este tipo de método, es la poda de los árboles de decisión en la que se escoge uno entre *N* atributos aleatoriamente [18], [72].

4.8 Comparación entre los métodos de ensamble

Los métodos de Boosting y Bagging fueron comparados con el uso de árboles de decisión y redes neuronales en [54]. En esa investigación, los autores obtuvieron que Boosting fue más preciso que Bagging, pero en problemas con ruido, la robustez de Bagging fue superior.

En [4] también se hizo un estudio comparativo de estos dos métodos para redes bayesianas y árboles de decisión, teniendo en consideración los resultados alcanzados, se obtuvo que Boosting, en general, tuvo un mejor comportamiento a través del algoritmo Adaboost.

Bagging, Boosting y RSM fueron evaluados en [17] concluyendo que Boosting fue el más preciso de los tres para los problemas sin ruido. En [4] se confirman los resultados obtenidos, ya que Bagging fue el mejor método para los problemas con ruido (datos faltantes o erróneos), mientras que Boosting presentó el peor comportamiento. Sin embargo, otros autores en [58] encontraron para árboles de decisión que el mejor de los tres fue RSM, alcanzando la mayor precisión. Por otro lado, el estudio llevado a cabo en [30] mostró, en trabajos relacionados con datos geográficos y de teledetección, que RSM fue mucho más rápido en el entrenamiento con relación a Boosting y Bagging. Una comparación general y resumida de todos los métodos descritos en epígrafes anteriores, se muestra en el Cuadro 2.

Cuadro 2. Diferencias entre métodos de ensamble.

Método	Manipulación de los datos	Uso de datos sin ruido	Uso de datos con ruido
Boosting	Ejemplos de entrenamiento	Mejores resultados	Terceros mejores resultados
Bagging	Ejemplos de entrenamiento	Segundos mejores resultados	Mejores resultados
Forest	Atributos de entrada	-	-
Cross-validated committees	Ejemplos de entrenamiento	-	-
Error-correcting output coding	Datos de salida	-	-
Randomization	Inclusión de aleatoriedad	Terceros mejores resultados	Segundos mejores resultados

4.9 Algunas soluciones a los problemas de los métodos de ensamble

La aplicación de varios clasificadores que combinan sus predicciones, a pesar de las ventajas descritas respecto a la respuesta de un único clasificador, posee algunos problemas que están siendo revisados y a los que se le han encontrado soluciones por parte de la comunidad científica.

Entre los problemas pueden citarse: el alto consumo de recursos en el entrenamiento de conjuntos de modelos, la necesidad de almacenar a todos los modelos del conjunto, la inversión de un considerable tiempo de respuesta y la pérdida de la claridad o comprensión del modelo.

No obstante, los errores que puede provocar uno o más clasificadores pueden ser neutralizados por la influencia de otros modelos.

4.9.1 Poda de árboles de decisión con Adaboost

En algunas aplicaciones, el número de árboles de decisión que se requiere para obtener una precisión razonable con el uso del método de Adaboost es elevado, a la vez que se consumen muchos recursos [23]. Por esta razón, en [56] se propuso la poda de este modelo Boosting. Los autores definieron el método de poda como un procedimiento que tiene en cuenta como entrada a un conjunto de datos de entrenamiento, el propio algoritmo AdaBoost y el tamaño máximo de memoria empleado por los clasificadores que participan. El principal objetivo de esta solución

es construir el mejor método ensamble posible que no sobrepase el máximo permitido de la memoria.

En los estudios realizados por [56], se concluyó que los métodos de ensamble producidos por AdaBoost pueden ser radicalmente podados (60-80%). Los mejores resultados de los métodos de poda empleados fueron los obtenidos por el método Poda de Kappa y el de Poda por Reducción del Error.

4.9.2 Miniboosting

Por otra parte, en [61] se estudió el método Boosting en el contexto de pequeñas uniones en árboles de decisión, mostrando que el proceso de reasignación de los pesos puede ser mejorado y los resultados de la combinación pueden representarse por un simple árbol de decisión. Este método conocido como Miniboosting, mejora el proceso de asignación de los pesos en los conjuntos de entrenamiento con pocos clasificadores. De modo que con dos clasificadores, el el resultado final será el emitido por el clasificador que tenga el valor inferior de error y por lo tanto, mayor votación por peso.

El estudio se realizó para tres clasificadores identificados como C_1, C_2 y C_3 cuya integración se denota por C_B; $P(e_1)$, $P(e_2)$, $P(e_3)$ y $P(e_B)$ representan sus respectivas probabilidades de error bajo la distribución de pesos original (uniforme) para los casos de entrenamiento. La misma notación se extiende para ejemplos más complejos; por ejemplo, $P(e_1e_2)$ denotará la probabilidad de los errores simultáneos de C_1 y C_2.

Como una simplificación, se usaron los pesos uniformes sugeridos por [8] en Bagging en lugar de los pesos definidos en Adaboost, ya que estos últimos dependen de los valores del error. Si existen tres o más clases, la combinación es posible si cada clasificador predice una clase diferente. En esta situación, la clase predicha por el clasificador original C_1 será la preferida.

De esta forma C_B clasificará de forma incorrecta un caso $\overline{e}$ sólo cuando al menos dos de los tres clasificadores que conforman el conjunto cumplen con:

$$P(e_B) \le P(e_1e_2\overline{e}_3) + P(e_1\overline{e}_2e_3) + P(\overline{e}_1e_2e_3) + P(e_1e_2e_3)$$

Aplicando la regla de independencia, cada término de la expresión anterior puede ser reemplazado por un producto de probabilidades más simples, por ejemplo:

$$P(e_1e_2\overline{e}_3) = P(e_1) \times P(e_2) \times (1 - P(e_3))$$

A menos que $P(e_1)$, $P(e_2)$, y $P(e_3)$ sean todas 0, el miembro derecho de la primera expresión, seguirá siendo estrictamente positivo.

Bajo el paradigma de Adaboost, los clasificadores en forma de árbol de decisión sucesivos parecen tener un error de solapamiento pequeño en el conjunto de entrenamiento. Cuando el número de clasificadores se limita a tres, sin embargo, es importante minimizar el error de solapamiento entre C_1 y C_3 como entre las parejas sucesivas C_1/C_2 y C_2/C_3. Los casos mal clasificados tanto por C_1 como por C_2 serán necesariamente clasificados de forma incorrecta por C_B. Estas observaciones sugieren que antes de que el tercer clasificador se construya se debe: poner en el valor 0 los pesos de cualquier caso mal clasificado tanto por C_1 y C_2, ya que el desempeño de C_3 en este caso es irrelevante y aumentar el peso de los casos mal clasificados tanto por C_1 o C_2 (pero no ambos), en lugar de sólo esos casos mal clasificados por C_2.

Miniboosting es un método simple para árboles de decisión (sólo 3 clasificadores), que incrementa de precisión de la clasificación.

4.9.3 Delegación de clasificadores

Recientemente se ha descrito el método de delegación de clasificadores [24], cuyo lema es "hacer que otros hagan lo que tú no puedes hacer bien".

La cobertura o soporte y la confianza son dos definiciones que se aplican a reglas de decisión para su evaluación. La cobertura es el número de instancias a las que se le aplica la regla y se predice correctamente y la confianza es la proporción de instancias que la regla predice correctamente, es decir, es el valor de la cobertura dividida por el número de instancias a las que se puede aplicar la regla [33].

En esta investigación también se ha introducido el concepto de clasificador prudente, que tiene como tarea clasificar sólo aquellos ejemplos para los cuales sus predicciones tienen alta confianza, dejando los ejemplos rechazados a otro clasificador.

Esto significa que este tipo de clasificador ofrece predicciones para un subconjunto de datos de entrada para el cual es más seguro que los otros, pero se abstiene para el resto de las entradas. Con la delegación de modelos en lugar de la combinación se evitan algunos de los problemas de los métodos de ensamble, en particular, la pérdida de la comprensibilidad y el uso excesivo de recursos computacionales.

Según los autores si un clasificador prudente $f^{(1)}$ decide que no es competente para clasificar un ejemplo con suficiente confianza, pero quiere completar el trabajo, en este caso puede delegar el ejemplo a otro clasificador. Si existe un segundo clasificador $f^{(2)}$ y un umbral de confianza τ, entonces la regla de decisión para delegar es:

$$\text{IF } f^{(1)}_{CONF}(e) > \tau \text{ THEN PREDICC } f^{(1)}_{CLASS}(e) \text{ ELSE PREDICC } f^{(2)}_{CLASS}(e)$$

Para cada clase *c* de un total de *C* clases, la función $f_{CLASS}(e)$ retorna la clase asignada por el clasificador *f* para el ejemplo *e* y la función $f_{CONF}(e)$ retorna la confianza de la predicción dada por el clasificador *f* para el ejemplo *e*.

Una manera natural de obtener el clasificador $f^{(2)}$ es entrenarlo sólo con los ejemplos de entrenamiento para que los que $f^{(1)}$ tiene baja confianza. De esta manera, el segundo clasificador se especializará para estos ejemplos. Más formalmente, si se tiene un conjunto de entrenamiento *Tr*, un clasificador *f* y un umbral de confianza τ, entonces se puede dividir este conjunto en dos juegos de datos: $Tr_f^{>} = \{e \in Tr : f_{CONF}(e) > \tau\}$ y $Tr_f^{\leq} = Tr - Tr_f^{>}$. Informalmente, en [24] se refieren a $Tr_f^{>}$ a los ejemplos "retenidos" o de "alta confianza" y $Tr_f^{\leq}$ como los ejemplos "delegados" o de "baja confianza".

Dado un fragmento ρ, un clasificador *f* y un conjunto de entrenamiento *Tr*, se puede obtener el umbral τ de la manera siguiente:

Es decir, τ es el mayor umbral de modo que por lo menos una proporción ρ de los ejemplos de entrenamiento tienen la más alta confianza. Con ello no se asegura una proporción ρ exactamente, porque puede haber muchos ejemplos con la misma confianza, pero se devuelve una aproximación superior a esta proporción. Este método se llama el Porcentaje Absoluto Global. La figura 15 ilustra el proceso de delegación de clasificadores.

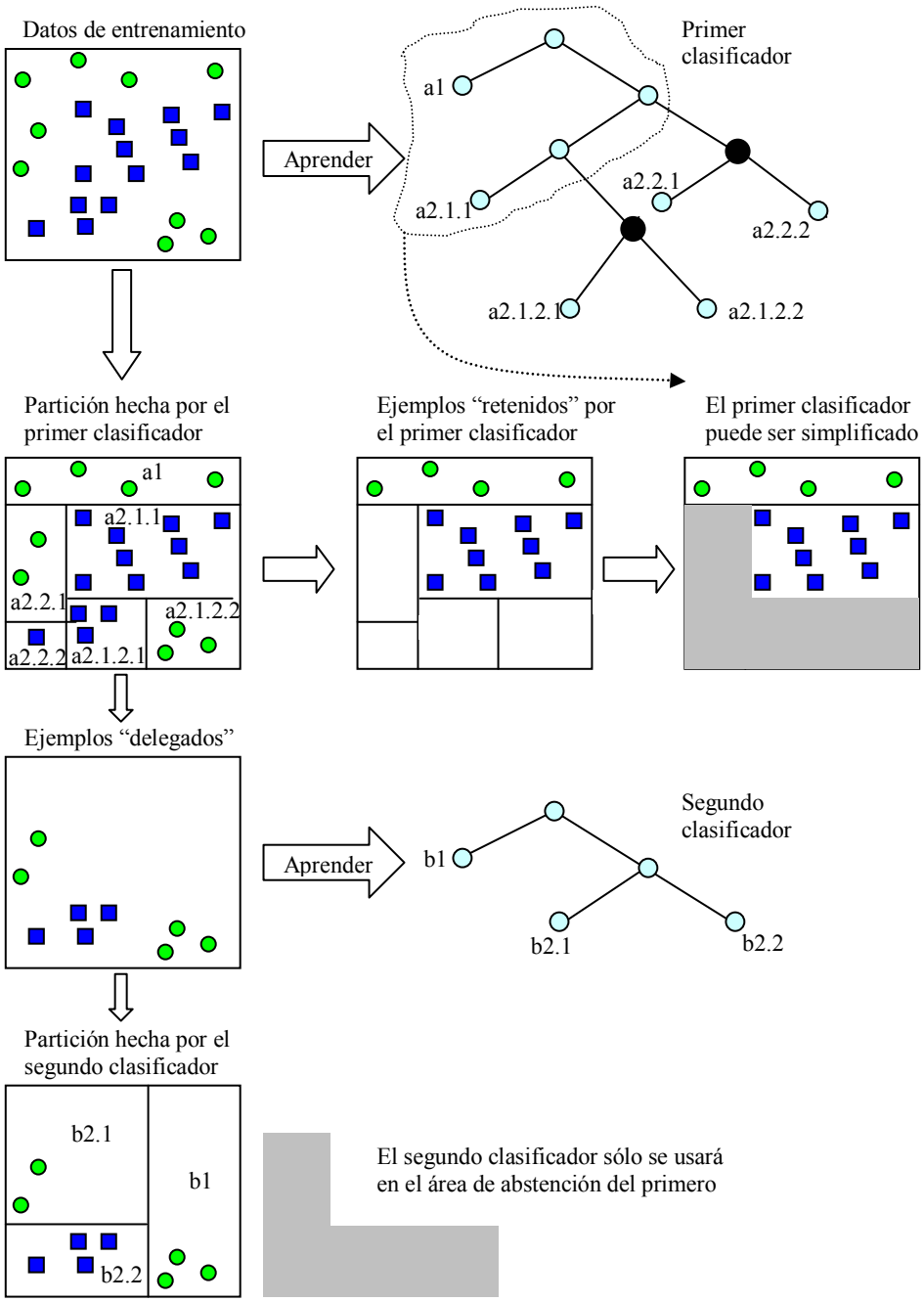

Figura 15. Delegación de clasificadores con árboles de decisión, tomado de [24].

Con este método no se combinan las predicciones de los clasificadores, cada instancia es clasificada por un clasificador, lo que evita que se degrade la comprensibilidad del modelo en comparación a los métodos de ensamble clásicos. También son más eficientes ya que el siguiente clasificador aprende usando un menor número de ejemplos respecto a su antecesor.

5 Métodos Híbridos

Los métodos híbridos a diferencia de los métodos de ensamble, combinan diferentes algoritmos de aprendizaje. Entre ellos: Stacking y Cascading.

5.1 Método de Acumulación (Stacking)

Stacking [83] combina múltiples clasificadores generados mediante distintos algoritmos de aprendizaje L_1, …, L_N para un mismo conjunto de datos S, el cual está formado por ejemplos $s_i=(x_i,y_i)$. En la primera fase, se genera un conjunto de clasificadores del nivel base C_1, C_2, ..., C_N, donde $C_i=L_i(S)$. En la segunda fase, un clasificador en el meta-nivel combina las salidas provenientes de los clasificadores del nivel base que trabajan en paralelo (figura 16).

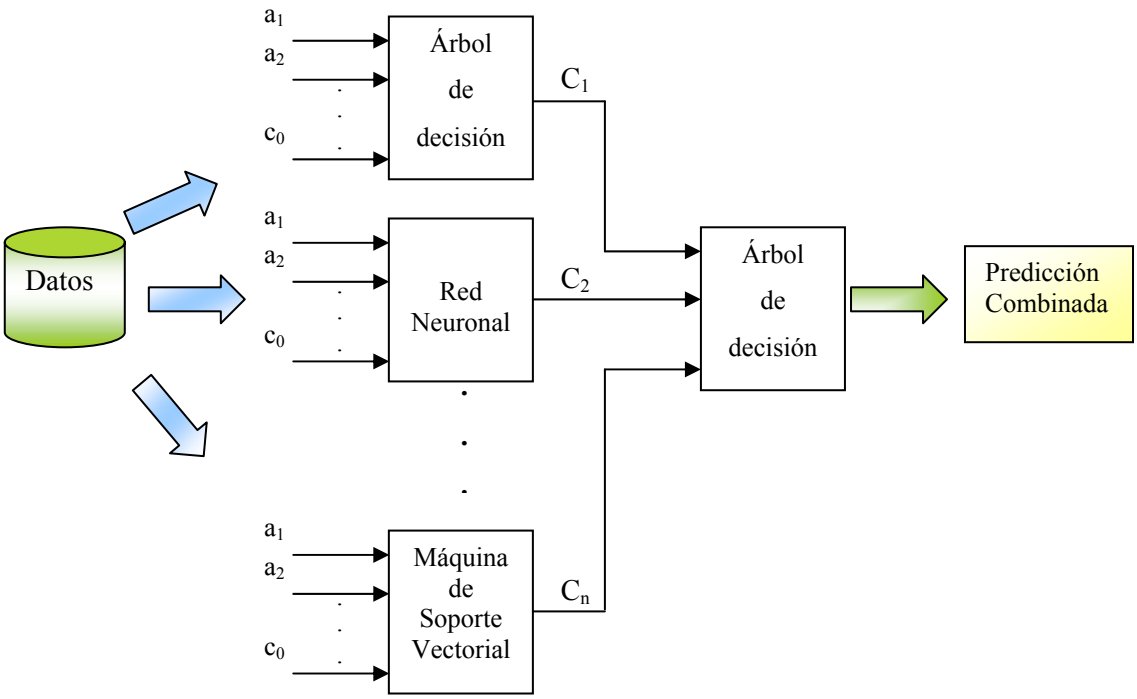

Figura 16. Método Stacking tomado de [33].

En la generación de un conjunto de entrenamiento para la etapa de meta-nivel se aplica un procedimiento *leave-one-out* o de validación cruzada [22], explicado en la sección 4.6.

La combinación de los clasificadores se realiza por el esquema de votación. El esquema de votación más simple es el voto de pluralidad. Cada clasificador del nivel de base emite un voto para su predicción. El ejemplo es clasificado en la clase que obtiene el mayor número de votos. Sin embargo, la votación mayoritaria funciona bien cuando todos los modelos combinados tienen una precisión aceptable, y en este caso no siempre se puede asegurar en todos los clasificadores [33].

En [77] los clasificadores del nivel base predicen también las distribuciones sobre el conjunto de los valores de la clase, en lugar de los simples valores de la clase. Los atributos del meta-nivel son las probabilidades de cada uno de los valores de la clases devueltas por cada uno de los clasificadores de nivel de base. Los autores defienden que esto les permite no sólo usar las predicciones, sino que también se puede emplear la confianza de los clasificadores del nivel de base.

Cada clasificador del nivel de base predice una distribución de probabilidad (DP) sobre los posibles valores de la clase. La predicción del clasificador de base C aplicado al ejemplo x será:

$$p^C(x) = \left(p^C(c_1|x), p^C(c_2|x), \ldots p^C(c_m|x)\right),$$

Donde $\{c_1, c_2, ..., c_m\}$ es el conjunto de posibles valores de la clase y $p^C(c_i|x)$ denota la probabilidad estimada (y predicha) por el clasificador *C* de que el ejemplo *x* pertenece a la clase c_i. La clase c_j con la mayor probabilidad $p^C(c_j|x)$ es la que predice el clasificador *C*. Los atributos del meta-nivel son las probabilidades predichas para cada posible clase por cada uno de los clasificadores del nivel base.

La regresión lineal multi-respuesta (RLM) se recomienda para el nivel de meta-aprendizaje. RLM es una adaptación de la regresión lineal. Para un problema de clasificación con *m* valores de clases $\{c_1, c_2, ..., c_m\}$, se formulan *m* problemas de regresión. Para cada clase c_j, se construye una ecuación lineal RL_j para predecir una variable binaria que tiene valor 1 si el valor de la clase es c_j y 0 en caso contrario. Dado un nuevo ejemplo *x* para clasificar, se calcula $RL_j(x)$ es calculado para todo *j*, y la clase *k* se predice con la máxima $RL_k(x)$.

El principal éxito de Stacking se debe a la capacidad de combinar las predicciones del conjunto de clasificadores.

La solución al problema de la votación mayoritaria es la introducción de una fase de meta-aprendizaje, que consiste en aprender cómo integrar los resultados a partir de múltiples algoritmos de aprendizaje. Para ello se usa un árbitro que es un clasificador que se entrena para resolver las discrepancias entre los clasificadores de la base.

El conjunto de entrenamiento para el árbitro se selecciona de todos los datos disponibles usando una regla de selección. Una regla de selección podría ser "Seleccionar los ejemplos cuya clasificación por parte de los clasificadores del nivel base no pudo ser predicha de forma consistente". El árbitro, junto con una regla de arbitraje, toma la decisión final basándose en las predicciones de los clasificadores del nivel base. Un ejemplo de una regla de arbitraje sería "Usar la predicción del árbitro cuando los clasificadores del nivel base no obtengan una mayoría" [27].

5.2 Método en Cascada (Cascading)

La idea básica de la generalización de Cascading o *Cascade Generalization* es utilizar secuencialmente un conjunto de clasificadores. En cada paso hay que realizar una extensión de los datos originales a través de la inserción de nuevos atributos. Los nuevos atributos se derivan de la distribución de la probabilidad de la clase dada por un clasificador de la base. Este paso constructivo amplía el lenguaje de representación para los clasificadores de alto nivel [27].

En [27] se define un operador constructivo $\varphi(\vec{x}, \mathrm{M})$ donde M representa al modelo *L(D)* para los datos de entrenamiento *D* y el clasificador *L*, mientras que $\vec{x}$ representa un ejemplo. Para el ejemplo $\vec{x}$ el operador φ concatena el vector de entrada $\vec{x}$ con la distribución de la probabilidad de la clase de salida. Si el operador φ se aplica a todos los ejemplos del conjunto de datos D' se obtiene un nuevo conjunto de datos D''. Cada ejemplo en $\vec{x} \in D''$ tiene un ejemplo equivalente en D', pero aumentado con *c* nuevos atributos, donde *c* representa el número de clases. Los nuevos atributos son los elementos del vector de distribución de la probabilidad de la clase obtenida al aplicar el clasificador *L(D)* al ejemplo $\vec{x}$.

La generalización de Cascading es una composición secuencial de clasificadores que en cada nivel aplica el operador Φ. Dado un conjunto de entrenamiento *E*, un conjunto de prueba *T*, y 2 clasificadores L_1 y L_2, la generalización de Cascading es como sigue, usando al clasificador L_1, que genera los datos del $Nivel_1$:

$$Entrenam.Nivel_1 = \Phi(E, \mathrm{A}(L(E), E))$$

$$TestNivel_1 = \Phi(T, \mathrm{A}(L(E), T))$$

El clasificador L_2 aprende con datos de entrenamiento del $Nivel_1$ y clasifica que los datos de prueba del $Nivel_1$:

$$\mathrm{A}(L_2(Entrenam.Nivel_1), TestNivel_1)$$

Estos pasos conforman la secuencia básica de una generalización de Cascading del clasificador L_2 posterior al clasificador L_1. Se representa la secuencia básica por el símbolo ∇. La composición anterior podría representarse sucintamente por:

$$L_2 \nabla L_1 = A(L_2(Entrenam.Nivel_1), TestNivel_1)$$

El principal éxito de Cascading reside en obtener un modelo que puede usar términos en la representación del lenguaje de los clasificadores de los niveles más bajos. Los mejores resultados se obtienen con un pequeño número de clasificadores de base.

El ejemplo desarrollado en [27] utiliza el conjunto de datos de la UCI *Monks-2* [76]. El conjunto de datos *Monks* describe el dominio de un robot artificial y es bien conocido por la comunidad de Sistemas de Aprendizaje. Los robots se describen a través de seis atributos y clasificados mediante dos clases. La regla de decisión para el problema es: "El robot va bien si dos de sus atributos poseen sus primeros valores".

Para este ejemplo se ha comparado el error en que incurre de forma individual dos clasificadores a través de la validación cruzada, uno por árbol de decisión (32.9%) y otro por *Naive Bayes* (34.2%) con la composición del modelo en Cascading por la acción conjunta de ambos.

Los datos del $Nivel_1$ se generan usando como *clasificador Naive Bayes*. Este clasificador construye un modelo a partir de los datos de entrenamiento originales. Este modelo se usa para calcular una distribución de probabilidad de las clases para cada ejemplo en el conjunto de entrenamiento y de prueba. El $Nivel_1$ se obtiene extendiendo los conjuntos de entrenamiento y de prueba con la distribución de la probabilidad de las clases calculada a través de *Naive Bayes*, donde los nuevos atributos *P(OK)* y *P(not_OK)* son las probabilidades de que los ejemplos pertenezcan a la clase "OK" y "not_OK", respectivamente. El árbol de decisión (C4.5) se entrena con los datos de entrenamiento del $Nivel_1$ y clasifica los datos de prueba en ese nivel. La composición de $C4.5 \nabla NaiveBayes$, obtiene un error de 8.9%, lo que es sustancialmente inferior a los errores individuales tanto de C4.5 como de *Naive Bayes*. En este caso, Cascading fue capaz de alcanzar un notable incremento de la precisión. La figura 17 se corresponde con el ejemplo antes descrito.

Existen varios aspectos que diferencian a Cascading de Stacking:

- Los nuevos atributos son continuos. Ellos toman la forma de una distribución de probabilidad de la clase. La combinación de clasificadores por medio de las clases categóricas pierde la fuerza del clasificador en su predicción. El uso de las distribuciones de probabilidad de la clase permite explorar esta información.
- Todos los clasificadores tienen acceso a los atributos originales. Cualquier nuevo atributo construido en las capas más bajas se considera exactamente de la misma manera que cualquiera de los atributos originales.
- La generalización de Cascading no usa validación cruzada internamente. Este aspecto afecta a la eficiencia computacional de este método.
- Stacking es paralelo por naturaleza, Cascading es secuencial.

Algunos ejemplos de los datos originales de entrenamiento:

head	body	smile	holding	color	tie	class
round	round	yes	sword	red	yes	not_OK
round	round	no	ballon	blue	no	OK

Nivel$_1$:

head	body	smile	holding	color	tie	P(OK)	P(notOK)	class
round	round	yes	sword	red	yes	0.135	0.864	not_OK
round	round	no	ballon	blue	no	0.303	0.696	OK

Uno de los árboles creados por C4.5∇NaiveBayes:

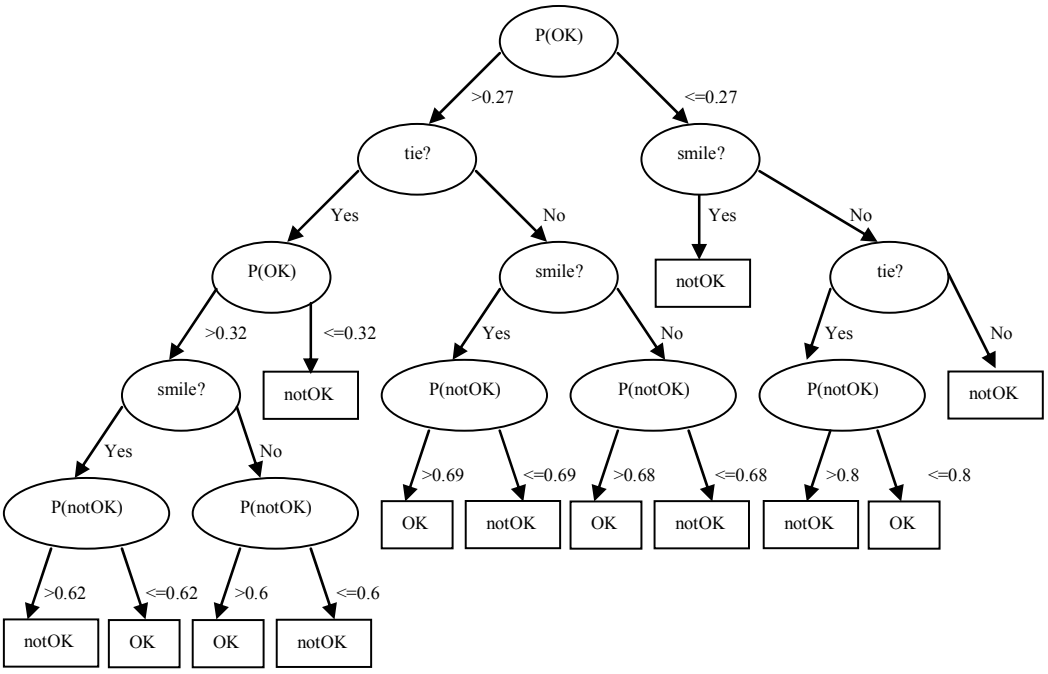

Figura 17. Ejemplo de Cascading, tomado de [27].

6 Métodos de no ensamble

6.1 Redes Neuronales de Funciones de Base Radial (RNFBR)

Las redes neuronales de funciones de base radial (RNFBR) constituyen un método de construcción que puede ser visto como un método de ensamble aditivo donde cada función de base individual forma una de las hipótesis.

Recientemente en las RNFBR se han hallado soluciones muy atractivas para muchos problemas de ingeniería. El incremento de la popularidad de las RNFBR se debe a su simple estructura topológica, ellas localmente colocan neuronas y tienen la habilidad de tener un rápido algoritmo de aprendizaje en comparación con las redes neuronales de alimentación hacia adelante multi-capa.

La estructura de una RNFBR se muestra en la figura 18, la cual posee una arquitectura similar a la tradicional red neuronal de alimentación hacia adelante con tres capas. La construcción de la RNFBR incluye 3 capas diferentes con la arquitectura de alimentación delantera. La capa de entrada de esta red es un conjunto de *n* unidades, las que aceptan los elementos de un vector *n-dimensional* de las características de entrada. Existen tantas neuronas como datos necesarios a transmitir. Las unidades de entrada están completamente conectadas a la capa oculta con *m* unidades ocultas. La capa de entrada solamente realiza una función simple de transmisión de datos.

La siguiente capa es la capa oculta a la que llegan los datos de la capa de entrada. El objetivo de la capa oculta es hacer un cluster de los datos y reducir su dimensionalidad. En esta estructura la capa oculta es llamada unidades de funciones de base radial [31]. La capa oculta aplica una transformación no lineal sobre los datos de entrada por medio de las funciones de base radial. La función más utilizada es la gaussiana aunque existen otras (multi-cuadráticas, multi-cuadráticas generalizadas, multi-cuadráticas inversas, multi-cuadráticas inversas generalizadas y cúbicas). Por último las salidas de la capa oculta se envían a las *r* neuronas de la capa de salida, multiplicadas por el peso de la conexión, y sumadas todas ellas, en definitiva una combinación lineal de las salidas de la capa oculta.

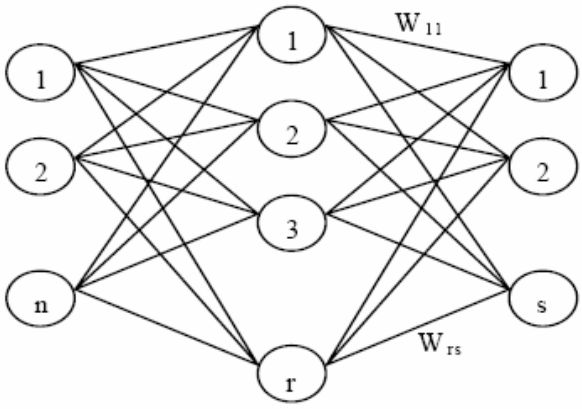

Figura 18. Estructura de la red neuronal con funciones de base radial.

La RNFBR es una clase de red neuronal, donde la función de activación de las unidades ocultas es determinada por la distancia entre el vector de entrada y un vector prototipo. La función de activación de las unidades de funciones de base radial se expresa por:

$$R_i(x) = R_i\left(\frac{\|x - c_i\|}{\sigma_i}\right), i = 1,2,...,m$$

donde x es un vector *n-dimensional* de entrada de las características, c_i es un vector *n-dimensional* llamado el centro de la unidad de la función de base radial, σ_i es el ancho de la unidad de la función de base radial y m es el número de neuronas en la capa oculta. Como ya se expresó generalmente la activación de las funciones de base radial es una función gaussiana con la media del vector c_i y la varianza del vector σ_i:

$$R_i(x) = \exp(-\frac{\|x - c_i\|^2}{\sigma_i^2})$$

σ_i^2 representa las entradas diagonales de la matriz de covarianza de la función gaussiana. Las unidades de salida son lineales y por lo tanto la respuesta de la *j-ésima* unidad de salida para la entrada *x* está dada por:

$$y_j(x) = \sum_{i=1}^{m} R_i(x) \times w_{ij}, \quad j = 1,...,r$$

donde w_{ij} es el peso de la conexión de la unidad de función de base radial *i-ésima* del nodo de salida *j-ésimo*.

6.2 Método de mezcla jerárquica de expertos

El modelo de mezcla jerárquica de expertos [41] está pensado para el aprendizaje supervisado en situaciones donde los datos de entrenamiento están siendo generados por una mezcla de expertos por separado. En una mezcla jerárquica, las hipótesis generales se combinan a través de una puerta de la red, la cual decide qué pesos deben ser empleados, basándose en las características de los datos. Por ejemplo, en un sistema de reconocimiento, se puede efectuar la tarea de distinguir la pronunciación de las palabras en idioma inglés: "bee", "tree", "gate", y "mate". Esta tarea se descompone en dos grandes subproblemas: distinguir "bee" de "tree" y distinguir "gate" de "mate" [15].

La figura 19 muestra la red probabilística para un simple modelo de mezcla de expertos para este caso. De acuerdo a este modelo, un ejemplo de entrenamiento (x_i,y_i) se genera primero por los datos de x_i, seguidamente se escoge un experto e_i estocásticamente (dependiendo del valor de x_i), y se selecciona entonces la clase y_i dependiendo de los valores de x_i y e_i.

Existen varios aspectos importantes a tener en consideración sobre este modelo. Primero, la dirección de causalidad se invierte desde una red *Naive Bayes*. Segundo, si se asume que todas las características serán siempre observadas, la distribución de la probabilidad *P(X)* en el nodo final del gráfico no es necesaria para el modelo. Tercero, a menos que se asuman algunas consideraciones de peso, la distribución de la probabilidad *P(Y|X,E)* será extremadamente compleja, porque esta debe especificar la probabilidad de las clases como una función de cada posible combinación de las características para *X* y el experto *E*.

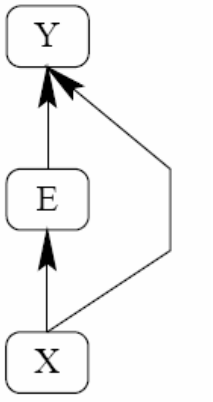

Figura 19. Modelo probabilística que describe una mezcla de expertos, tomado de [15].

En este modelo se asume que cada distribución de la probabilidad presenta una forma sencilla (figura 20). Cada valor de la variable aleatoria *E* especifica a un experto o clasificador diferente. Las características de entrada *x* se alimentan dentro de cada uno de los expertos y dentro de una puerta de la red o *gating network*. La salida de cada experto es una distribución de probabilidad de los expertos. Este modelo general tiene la siguiente forma analítica:

$$P(y|x) = \sum_{e} g_e(x) p_e(y|x)$$

donde el índice *e* varía según los distintos expertos. El valor $g_e(x)$ es la salida de la puerta de la red del experto *e*. El valor $p_e(y|x)$ es la distribución de la probabilidad sobre las distintas salidas de clases por el experto *e*.

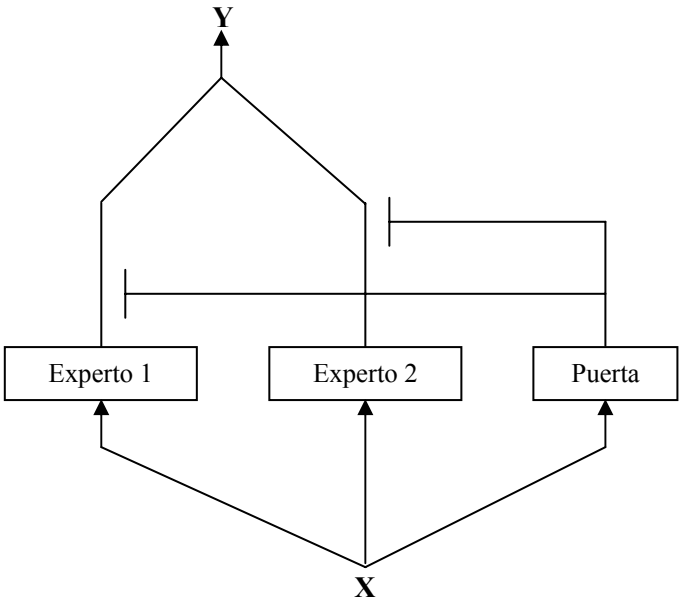

Figura 20. Modelo de mezcla de expertos visto como una red neuronal especializada, tomado de [15].

7 Caso de estudio

Durante el desarrollo de este estudio se siguieron los pasos del proceso de minería de datos definidos en [12]. El objetivo es tomar como caso de estudio los datos procedentes de las variables empleadas en la emisión de las categorías de aptitud física de las tierras del cultivo de la caña de azúcar en Cuba para aplicar técnicas de minería de datos que, a través de clasificadores, permitan predecir las categorías de aptitud de la tierra (cuatro categorías) a través de factores del suelo, del clima y agrícolas, presentes en las parcelas estudiadas. Una vez que se implementen clasificadores individuales precisos, se probará que con el uso de multiclasificadores es posible aumentar la precisión.

Los datos del estudio proceden del Instituto Nacional de Investigaciones de la Caña de Azúcar (INICA), institución adjunta al Ministerio del Azúcar de Cuba, responsable de las investigaciones que desde 1999 se desarrollan para la diversificación de la agricultura cañera, tomando como base la evaluación de la aptitud física de las tierras cultivadas con este fin.

La base de datos utilizada posee 1000 registros que corresponden a parcelas cultivadas con caña de azúcar. Las variables que las componen son 12 factores asociados al suelo, al clima y características agrícolas (pendiente del terreno, pedregosidad, rocosidad, salinidad, acidez del suelo, capacidad de intercambio catiónico, drenaje, compactación, precipitaciones, profundidad

efectiva, agrupamiento agroproductivo del suelo y categoría de aptitud de la tierra). De ellos existen 2 atributos numéricos y 10 nominales.

El valor del atributo a predecir en este trabajo está representado en la etiqueta "eval", perteneciente a la aptitud física de las tierras para el cultivo de la caña de azúcar.

En el preprocesamiento de los datos para asegurar la calidad de los mismos se utilizó *Mineset* de *Silicon Graphics, Inc*., debido a que la capacidad de visualización de este sistema en relación a *WEKA* es más detallada e ilustrativa.

A continuación se muestra, a través de un conjunto de datos, la aplicación de algunas de las técnicas explicadas en este trabajo. En la investigación se utilizó la herramienta *WEKA* (*Waikato Environment for Knowledge Analysis*) 3.4 de la Universidad de Waikato, software que se encuentra de forma gratuita en el sitio oficial de esta institución en Internet y contiene más variedad de algoritmos en comparación con *Mineset* para la aplicación de técnicas supervisadas y no supervisadas.

En *Mineset* se utilizó la opción *Statistics Viewer* para visualizar si existía alguna variable cuyos valores estaban significativamente lejos de los valores del conjunto. La visualización de la distribución de valores se obtuvo a través de histogramas, en el caso de los atributos nominales y gráficos de cajas (*boxplots*) en el caso de los atributos numéricos (figura 21). Los gráficos muestran que los datos de las variables no presentan ruido.

Únicamente en el caso de la variable numérica "prfu" correspondiente a la profundidad efectiva del suelo, se observa que la media se encuentra alejada del valor máximo. 78 valores de un total de 1000 son anormalmente altos, sin embargo no se deben a errores humanos sino que en Cuba, los suelos dedicados al cultivo de la caña de azúcar en general no son muy profundos, pero existen zonas en las que pueden alcanzarse valores altos de profundidad efectiva (muy profundos) y en otros muy pequeños. Todos los valores de los atributos se corresponden con los datos reales establecidos para las parcelas que conforman el dominio de los datos, lo que significa que en su registro no se han cometido errores. Aunque atributos como "ph", "cic", "sali" y "agrup", no están uniformemente repartidos entre las distintas categorías, no indica que los valores que representan las categorías con menos porcentaje son equivocaciones, sino que como ya se ha mencionado son datos ciertos.

Se realizó además, un diagrama de sectores (figura 22) para observar el comportamiento de la variable "eval", que es la etiqueta en este problema. Como se muestra, los datos de la aptitud de las tierras están bien repartidos entre las cuatro categorías posibles.

De acuerdo a los resultados de los análisis estadísticos efectuados al conjunto de datos, se puede afirmar que los datos poseen alta calidad pues no son erróneos, son fiables, no hay faltantes y siguen un comportamiento al esperado en cada uno de los atributos, es decir, los datos no tienen ruido.

En la figura 23 se muestra a través de la utilidad *Explorer* de *WEKA* la composición de la base de datos y el número de registros por categoría de aptitud de la tierra. Esta variable posee los valores "A1" para las áreas sumamente aptas, "A2" para las moderadamente aptas, "A3" para las marginalmente aptas y "N" como áreas no aptas para el cultivo de la caña de azúcar.

Con la herramienta *WEKA* se probaron varios algoritmos de clasificación para seleccionar aquellos que presentasen alta precisión en la predicción de la categoría de la aptitud física de las tierras y determinar qué parcelas son aptas (A1, A2 y A3) y cuáles no (N), para posteriormente formar multiclasificadores y comprobar el incremento de la precisión con el uso de estas técnicas de combinación en relación a los resultados obtenidos utilizando clasificadores individuales.

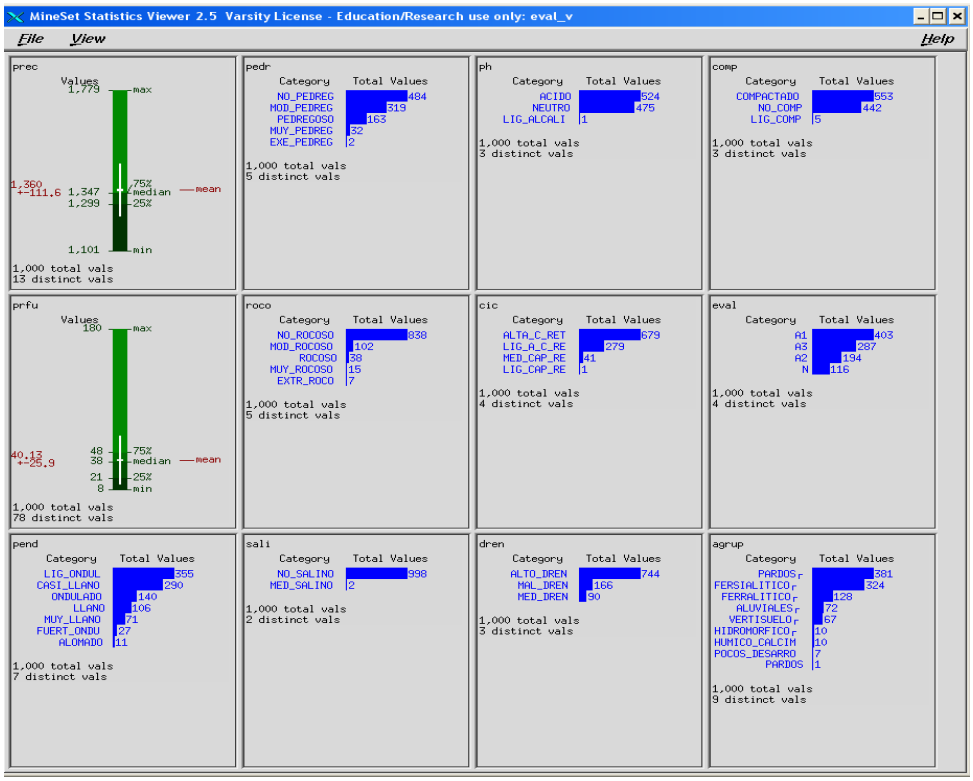

Figura 21. Visualización del comportamiento estadístico de las variables.

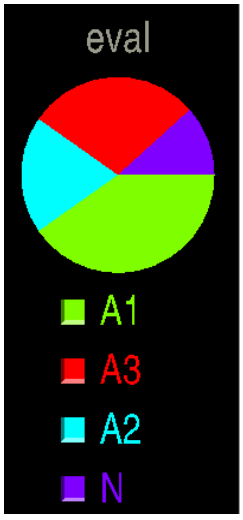

Figura 22. Gráfico de sectores para la etiqueta "eval".

La herramienta *WEKA* posee un gran número de métodos de aprendizaje, incluyendo métodos de ensamble e híbridos.

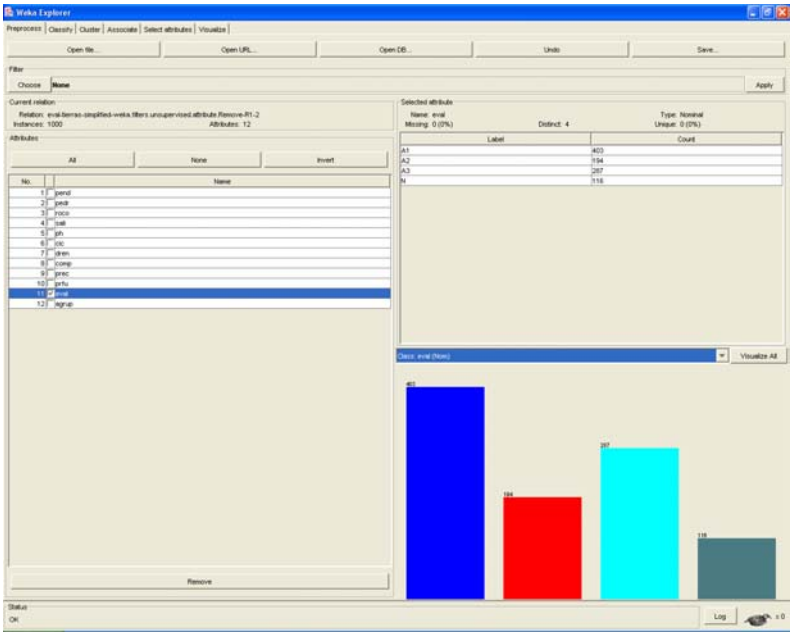

Figura 23. Composición de la base de datos de estudio a través de WEKA y visualización del número de registros en función de la categoría de aptitud de las tierras.

Se utilizaron tres métodos simples: árboles de decisión, aprendizaje bayesiano y aprendizaje basado en el vecino más cercano. El Cuadro 3 refleja los valores de la precisión de los tres métodos mencionados. Se ejecutaron un total de diez ejecuciones de validación cruzada de diez particiones.

Cuadro 3. Valores de precisión de tres métodos simples de aprendizaje.

Árbol de decisión	Vecino más cercano	Aprendizaje bayesiano
90,6%	83,2%	69,80%

Los resultados indican que el mejor método para este conjunto de datos es el árbol de decisión, después el aprendizaje a través del vecino más cercano y en tercer lugar, el aprendizaje bayesiano.

A continuación se construyeron multiclasificadores empleando los métodos de ensamble Bagging y Adaboost (Boosting) con los árboles de decisión, que es el inductor de mayor precisión individual, y comprobar de esta forma que se incrementa la precisión con respecto al uso de esta técnica de aprendizaje de forma individual. El Cuadro 4 muestra los valores obtenidos por los dos multiclasificadores. Ambos métodos permiten aumentar la precisión, Boosting fue superior a Bagging para este problema.

Cuadro 4. Valores de precisión para árboles de decisión de Bagging y Boosting.

Árboles de decisión	Bagging	Boosting
90,6%	92,2%	93,90%

La figura 24 muestra el comportamiento de la precisión en relación con el número de iteraciones, tanto para Boosting como para Bagging utilizando árboles de decisión. El aumento de la precisión para ambos multiclasificadores no es constante respecto al número de iteraciones, el salto mayor ocurre entre 0 y 20 iteraciones y los mayores valores los alcanza con 40.

Con el desarrollo de este ejercicio se corroboran los resultados obtenidos por [17], es decir, que para conjuntos de datos sin ruido, Boosting alcanza una mayor precisión que Bagging.

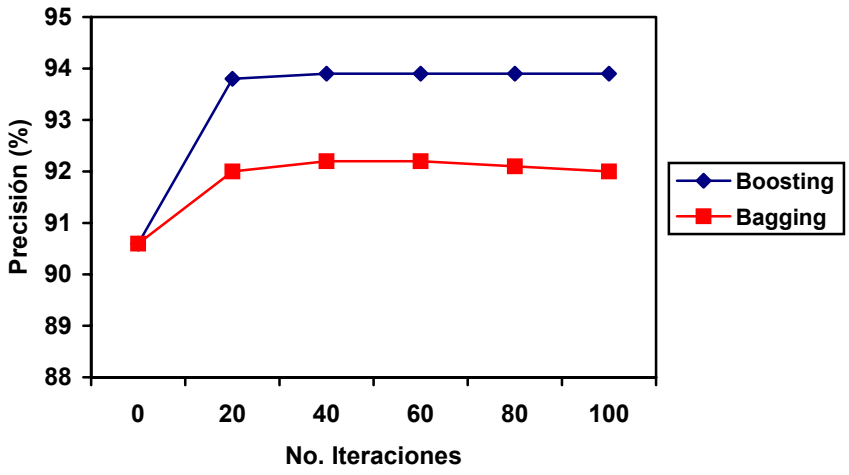

Figura 24. Precisión obtenida por Bagging y Boosting con árboles de decisión en relación con el número de iteraciones.

Al implementar Bagging y Boosting con el aprendizaje bayesiano, que fue el algoritmo que de manera individual obtuvo menor precisión, los resultados son similares, Boosting supera a Bagging (71.3% y 71%, respectivamente).

WEKA proporciona el método Stacking como método híbrido. En la implementación de este método, las técnicas de aprendizaje que se utilizaron para construir el modelo fueron los tres algoritmos iniciales (árboles de decisión, aprendizaje bayesiano y aprendizaje basado en el vecino más cercano) y para aprender el meta-modelo se usó *Naive Bayes*. Los resultados obtenidos reflejan un aumento de la precisión de Stacking en comparación a los clasificadores individualmente, véase el Cuadro 5.

Cuadro 5. Valores de precisión de Stacking y tres métodos simples de aprendizaje.

Árboles de decisión	Vecino más cercano	Aprendizaje bayesiano	Stacking
90,6%	83,2%	69,80%	91,2%

8 Conclusiones

Los multiclasificadores tienen muchas aplicaciones en áreas diversas. La combinación apropiada de dos o más clasificadores puede proporcionar una predicción más robusta, más confiable y eficiente que el uso de un único clasificador.

Existen diferencias en las formas de combinación y los resultados finales de los multiclasificadores dependiendo de los algoritmos y de las técnicas de aprendizaje utilizados. El uso de multiclasificadores introduce diversos criterios de clasificación que ofrecen mayor flexibilidad en la decisión final.

La combinación de la hipótesis de los multiclasificadores es un ejemplo del problema más general y más fundamental de la integración de la información por múltiples fuentes. La mayor significación de los multiclasificadores radica, en general, en que aumentan la precisión en

comparación con la que se obtiene con un solo clasificador, reducen el problema de sobreajuste y por lo general, evitan la selección de un modelo extenso.

No obstante, la propia estrategia de la combinación implica que algunos problemas introducidos por los multiclasificadores puedan ser disminuidos o resueltos, de ahí que existan unas combinaciones mejores que otras y que la comunidad científica de los sistemas de aprendizaje y de minería de datos realice continuos esfuerzos en lograr nuevas soluciones que superen los resultados ya alcanzados.

En el informe se incluye un caso de estudio consistente en un problema de clasificación para datos correspondientes a la evaluación de la aptitud física de las tierras dedicadas al cultivo de la caña de azúcar en Cuba. Se desarrollaron las distintas etapas correspondientes al proceso de minería de datos, para ello se utilizaron dos herramientas informáticas (*Mineset* y *WEKA*). Con *Mineset* se demostró a través de técnicas estadísticas y de visualización que los datos eran fiables y no erróneos. *WEKA* permitió realizar el procesamiento de clasificación.

La naturaleza de los algoritmos de aprendizaje influye en la precisión de la clasificación de un mismo conjunto de datos. En la investigación se utilizaron tres técnicas de aprendizaje distintas y de manera individual se obtuvieron clasificadores precisos. Al emplear métodos de ensamble (Boosting y Bagging) (una misma técnica de aprendizaje) se logró incrementar el valor de la precisión respecto al clasificador individual con el mismo aprendizaje.

Boosting obtuvo mejores resultados que Bagging, lo que ratifica otros estudios desarrollados, donde un comportamiento similar ocurre para conjuntos de datos exentos de ruido.

Se utilizó un método híbrido (Stacking), que combinó a las tres técnicas de aprendizaje y logró incrementar el valor de la precisión en comparación a cualquiera de las técnicas de aprendizaje de forma individual.

Referencias

[1] Al-Ani, A., Deriche, M.: A New Technique for Combining Multiple Classifiers using the Dempster-Shafer Theory of Evidence. Journal of Artificial Intelligence Research 17, 333-361, 2002.

[2] Alpaydin, E.: Voting over multiple condensed nearest neighbors. Artificial Intelligence Review. Kluwer Academic Publishers, The Netherlands, 11, 115-132, 1997.

[3] Battini, R., Colla, A.M.: Democracy in neural nets: voting schemes for classification. Neural Networks, vol. 7, 2, 691-707, 1994.

[4] Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting and variants. Machine Learning, Kluwer Academic Publishers, Boston, Manufactured in The Netherlands, vol. 36, 105–139, 1999.

[5] Baykut, A., Erçil, A.: Towards automated classifier combination for patter recognition. In: T. Wideatt, Fabio Roli (eds.): Multiple Classifier Systems. Springer Verlag, 94-105, 2003.

[6] Bose, R.C., Ray-Chaudhuri, D.K.: On a class of error-correcting binary group codes. Information and Control, 3, vol. 1, 68-79, 1960.

[7] Brand, J.D., Mason, J.S.D., Colomb, S.: Visual Speech: A Physiological or Behavioural Biometric? Lecture Notes in Computer Science, Springer-Verlag GmbH, vol. 2091, 157-168, 2001.

[8] Breiman, L.: Bagging predictors. Machine Learning, Kluwer Academic Publishers, Boston, Manufactured in The Netherlands, vol. 24, 2, 123-140, 1996.

[9] Breukelen, M., Duin, R.P.W., Tax, D.M.J., Hartog, J.E.: Handwritten digit recognition by combined classifiers. Kybernetika, vol. 34, 4, 381-386, 1998.

[10] Brunelli, R., Falavigna, D.:. Person Identification Using Multiple Cues. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, 10, 955-966, 1995.
[11] Bruzzone, L., Cossu, R.: A multiple-cascade-classifier system for a robust and partially unsupervised updating of land-cover maps. IEEE Transactions on Geoscience and Remote Sensing, vol. 40, 9, 1984-1996, 2002.
[12] Cabena, P., P. Hadjinian, R. Stadler, J. Verhees and A. Zanasi, Discovering Data Mining. From Concept to Implementation, Prentice Hall, 1998.
[13] Cheng, J., Greiner, R.: Learning Bayesian Belief Network Classifiers: Algorithms and System. Proceedings of 14 Biennial conference of the Canadian society for computational studies of intelligence, 2001.
[14] Cho, S.B., Kim, J.: Combining multiple neural networks by fuzzy integral and robust classification. IEEE Transactions on Systems, Man, and Cybernetics, 25, 380-384, 1995.
[15] De Borda, J.C.: Memoire sur les Elections au Scrutin. Historie de l'Academie Royale des Sciences, Paris, 1781.
[16] Dietterich, T. G.: Machine Learning Research: Four Current Directions AI Magazine. 18 (4), 97-136, 1997.
[17] Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning, Kluwer Academic Publishers, Boston, Manufactured in The Netherlands, vol. 40, 2, 139-157, 2000.
[18] Dietterich, T.G.: Ensemble Learning. In The Handbook of Brain Theory and Neural Networks, Second edition, (M.A. Arbib, Ed.), Cambridge, MA: The MIT Press, 405-408, 2002.
[19] Dietterich, T.G., Bakiri, G.: Solving multi-class learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, vol. 2, 263-286, 1995.
[20] Drucker, H., Cortes, C., Jackel, L.D., LeCun, Y., Vapnik. V.: Boosting and other ensembles methods. Neural Computation, vol. 6, 1289-1301, 1994.
[21] Dudoit, S., Fridlyand, J., Speed, T.P.: Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. Journal of the American Statistical Association, vol. 97, 457, 77-87, 2002.
[22] Džeroski, S., Ženko, B.: Is Combining Classifiers with Stacking Better than Selecting the Best One? Machine Learning. Kluwer Academic Publishers, Manufactured in The Netherlands, 54, 255-273, 2004.
[23] Ferri, C.: Multiclasificadores en minería de datos. Dep. de Sistemes Informàtics i Computació, Universitat Politècnica de València, València, Spain. Reunión Red Minería de Datos, Madrid, 2004.
[24] Ferri, C., Flach, P., Hernández-Orallo, J.: Delegating classifiers. Proceedings of the 21st Internacional Conference on Machine Learning (ICLM'04), 2004.
[25] Freitag, D., Kachitesllum, A.: Information extraction with HMMs and shrinkage. Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction, 1999.
[26] Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. Proceedings 13th International Conference on Machine Learning, 148-156, 1996.
[27] Gama, J., Brazdil, P.: Cascade Generalization. Machine Learning, Kluwer Academic Publishers, Boston, Manufactured in The Netherlands, vol. 41, 3, 315-343, 2000.
[28] Giacinto, G.: Design of Multiple Classifier Systems. Dottorato di ricerca in Ingegneria dell'Informazione, Elettromagnetismo applicato e Telecomunicazioni. Università degli Studio de Salermo, Italy, 1998.
[29] Giacinto, G., Roli, F. Ensembles of Neural Networks for Soft Classification of Remote Sensing Images. Proceedings of the European Symposium on Intelligent Techniques, March 20-21, Bari, Italy, 166-170, 1997.

[30] Gislason, P.O.; Benediktsson, J.A.; Sveinsson, J.R.: Random Forest classification of multisource remote sensing and geographic data. Proceedings of IEEE International Geoscience and Remote Sensing Symposium, IGARSS'04, vol. 2, 1049-1052, 2004.
[31] Haddadnia, J., Faez, K., Ahmadi, M.: N-feature neural network human face recognition. Image and Vision Computing. 22(12), 1071-1082, 2004.
[32] Hansen, L.K., Salamon, P.: Neural network ensemble. IEEE Transactions Pattern Analysis Machine Intelligence, vol. 12, 10, 993-1001, 1988.
[33] Hernández-Orallo, J., Ramírez, M.J., Ferri, C.: Introducción a la minería de datos. Pearson Educación, S.A., Madrid, 2004.
[34] Heute, L.: Combinaison de Classifieurs. Méthodes pour la Construction d'Ensembles de Classifieurs. Université de Rouen, France, 2005. http://www.univ-rouen.fr/psi/heutte/rdf/MethodoMCS.pdf
[35] Ho, T.K.: The random subspace method for constructing decision forest. IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 20, 8, 832-844, 1998.
[36] Hocquenghem, A.: Codes corecteurs d'erreurs. Chiffres, 2, 147-156, 1959.
[37] Huang, Y.S., Suen, C.Y.: A method for combining multiple experts for the recognition of unconstrained handwitten numerals. IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 17, 90-93, 1995.
[38] Jacobs, R.: Methods for combining experts' probability assessments. Neural Computation, 7, 867–888, 1996.
[39] Jain, A.K., Duin, R.P.W., Mao, J.:. Statistical Pattern Recognition: A Review. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, 1, 4-37, 2000.
[40] Jain, A.K., Pankanti, S., Prabhakar, S., Hong, L., Ross, A., Wayman, J.L: Biometrics: A Grand Challenge. Proceedings of ICPR, Cambridge, United Kingdom, 2004.
[41] Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. Neural Computation, 6(2), 181-214, 1994.
[42] Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, 3, 226-239, 1998.
[43] Kolen, J.F., Pollack, J.B.: Back propagation is sensitive to initial conditions. In Advances in Neural Information Processing Systems, Morgan Kaufmann, vol. 3, 860-867, San Francisco, CA, 1991.
[44] Krum, D., Omoteso, O., Ribarsky, W., Starner, T., Hodges, L.: Evaluation of a Multimodal Interface for 3D Terrain Visualization. IEEE Visualization, October 27-November 1, 2002.
[45] Kuncheva, L.I.: A theorical study on six classifier fusion strategies. IEEE Transactions on PAMI, 24 (2), 281-286, 2002.
[46] Kuncheva, L.I., Bezdek, J.C., Sutton, M.A.: On combining multiple classifiers by fuzzy templates. Conference of the North American, 193-197, 1998.
[47] Kuncheva, L.I., Jain, L.C.: Designing classifier fusion systems by genetic algorithms. IEEE Transactions on Evolutionary Computation, vol. 4, 4, 327-336, 2000.
[48] Kuncheva, L.I., Whitaker, C.J.: Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. Machine Learning. Kluwer Academic Publishers, The Netherlands, 51, 181-207, 2003.
[49] Last, M., Bunke, H., Kandel, A.: A feature-based serial approach to classifier combination. Pattern Analysis & Applications, 5. Springer-Verlag, London, 385-398, 2002.
[50] Lawrence, S., Lee, C., Tsoi, A.C., Back, A.D.:. Face Recognition: Convolutional Neural Network Approach. IEEE Transactions on Neural Networks, vol. 8, 1, 98-113, 1997.
[51] Lee, D.S., Srihari, S.N.: A theory of classifier combination: the neural networks approach. Proceedings 3rd International Conference on Document Analysis Recognition. IEEE Computer Society Press, vol. 1, 42-45, Montreal, Canada, 1995.
[52] Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Information and Computation, 108, 212-261, 1994.

[53] Lynch, R.S., Willet, P.K.: Classifier fusion results using various open literature data sets. IEEE International Conference on Systems, Man and Cybernetics, vol. 1, 723-728, 2003.
[54] Maclin, R., Opitz, D.: An empirical evaluation of bagging and boosting. Proceedings of the Fourteenth National Conference on Artificial Intelligence, Cambridge, MA. AAAI Press/MIT Press, 546-551, 1997.
[55] Maclin, R., Shavlik, J.: Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.
[56] Margineantu, D., Dietterich, T.: Pruning adaptive boosting. Proceedings of the Fourteenth International Conference on Machine Learning San Francisco, CA. Morgan Kaufmann, 211-218, 1997.
[57] Osadciw, L., Varshney, P., Veeramachaneni, K.: Improving Personal Identification Accuracy Using Multisensor Fusion for Building Access Control Applications. Proceedings of the Fifth International Conference on Information Fusion, July, Annapolis, Maryland, 2002.
[58] Pal, M.: Random forests for land cover classification. Proceedings of the International Geoscience and Remote Sensing Symposium, IGARSS '03, vol. 6, July 21-25, 3510-3512, 2003.
[59] Pankanti, S., Prabhakar, S., Jain, A.: On the Individuality of Fingerprints. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Hawaii, December, 805-812, 2001.
[60] Parmanto, B., Munro, P. W. and Doyle, H. R.: Improving committee diagnosis with resampling techniques. Advances in Neural Information Processing Systems 8, eds. Touretzky, D. S., Mozer, M. C. and Hesselmo, M., MIT Press, USA, 882-888, 1996.
[61] Quinlan, J.R.: MiniBoosting Decision Trees. Submitted to Journal of Artificial Intelligence Research, 1999.
[62] Rahman, A.F.R., Fairhurst, M.C.: Multiple classifier decision combination strategies for character recognition: A review. International Journal on Document Analysis and Recognition, 5. Springer-Verlag, GmbH, 166-194, 2003.
[63] Ranawana, R., Palade, V.: A Neural Network Based Multi-Classifier System for Gene Identification in DNA Sequences. Neural Computing and Applications Springer-Verlag, in printing.
[64] Raudys, S., Roli, F.: The Behavior Knowledge Space Fusion Method: Analysis of Generalization Error and Strategies for Performance Improvement. 4th Int. Workshop on Multiple Classifier Systems (MCS 2003), Guildford, United Kingdom, June 11-13, T. Windeatt and F. Roli Eds., LNCS 2709, 55-64, 2003.
[65] Raviv, Y., Itrator, N.: Bootstrapping with Noise: An Effective Regularization Technique. Connection Science Special Issue on combining estimators, vol. 8, 356-372, 1996.
[66] Roli, F.: Fusion of Multiple Pattern Classifiers. 8th National Conference of the Italian Association on Artificial Intelligence, September, Pisa, Italy, 2003.
[67] Sharkey, A.J.C.: Multi-net systems in combining artific neural nets: ensemble and modular multi-nets systems. Perspectives in neural computing. Springer, Berlin Heidelberg New York, 1999.
[68] Sharkey, A., Sharkey, N.: How to improve the reliability of artificial neural networks. Technical Report Cd-95-11, Department of Computer Science, University of Sheffield, 1995.
[69] Skalak, D.B.: Prototype selection for composite nearest neighbor classifiers. Technical Report 95-74, Department of Computer Science, University of Massachusetts, Amherst, MA, 1995.
[70] Skurichina, M., Duin, R.P.W.: Bagging for lineal classifiers. Pattern Recognition, vol. 37, 7, 909-930, 1998.

[71] Skurichina, M., Duin, R.P.W.: Bagging, Boosting and the Random Subspace Method for Linear Classifiers. Pattern Analysis & Application, Springer-Verlag, London, 5, 121-135, 2002.
[72] Smirnov, E.N.: Ensembles of Classifiers. Department of Computer Science, Maastricht University, Maastricht, The Netherlands.
http://www.cs.unimaas.nl/~postma/ml/Ensembles-2004.ppt
[73] Smyth, P.: Bounds on the mean classification error rate of multiple expert. Pattern Recognition Letters, in press, 1995.
[74] Stone, M.: Cross-Validatory Choice and Assessment of Statistical Predictions. Journal of the Royal Statistical Society 36, 111-147, 1974.
[75] Tahano, H., Keller, J.M.: Information fusion in computer vision using the fuzzy integral. IEEE Transactions on Systems, Man and Cybernetics, vol. 20, 3, 733-741, 1990.
[76] Thrun, S.B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Džeroski, S., Fahlman, S.E., Fisher, D., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R.S., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J., Zhang, J.: The Monk's Problems: A Performance Comparison of Different Learning Algorithms. Technical Report, CS-91-197, Pittsburgh, PA, 1991.
[77] Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research, 10, 271-289, 1999.
[78] Toh, K.A., Yau, W.Y.: Combination of hyperbolic functions for multimodal biometrics data fusion. IEEE Transactions on Systems, Man and Cybernetics, Part B. vol. 34, 2, 1196-1209, 2004.
[79] Tsymbal, A., Pecherizkiy, M., Puuronen, S., Patterson, D.W.: Dynamic Integration of Classifiers in the Space of Principal Components. ADBIS, 278-292, 2003.
[80] Van Erp, M., Schomaker, L.: Variants of the Borda count method for combining ranked classifier hypotheses. In Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, Amsterdam, September, 443-452, 2000.
[81] Vicente, M.A., Fernández, C.: Teoría de la decisión de Bayes. Curso Inteligencia Artificial y Reconocimiento de Patrones. Universitas Miguel Hernández, 36-38, 2004.
[82] Webb, G.I.: Multiboosting: a technique for combining Boosting and Wagging. Machine Learning. Kluwer Academic Publishers, The Netherlands, 40, 159-196, 2000.
[83] Wolpert, D.H.: Stacked Generalization. Neural Networks, Pergamon Press, vol. 5, 241-259, 1992.
[84] Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems Man and Cybernetics, 22(3), 418-435, 1992.
[85] Yan, W., Goebel, K.: Designing Classifier Ensembles with Constrained Performance Requirements, Proceedings of SPIE Defense & Security Symposium, Multisensor Multisource Information Fusion: Architectures, Algorithms, and Applications, 2004.
[86] Zhang, B., Srihari, S.N.: Class-wise multi-classifier combination based on Dempster-Shafer theory. Proceedings of the Seventh International Conference on Control, Automation, Robotics and Vision (ICARV 2002), Singapore, Dec 2-5, 2002.