CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

REPORTE TÉCNICO

Reconocimiento de Patrones

SERIE AZUL

RNPS No. 2142
ISSN Solicitado

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

SERIE AZUL

REPORTE TÉCNICO

Reconocimiento de Patrones

Selección y construcción de objetos para el mejoramiento de un clasificador supervisado: un análisis crítico.

Milton García Borroto, Yenny Villuendas Rey, Miguel Ángel Medina Pérez, Juliett Martínez López, José Ruiz Shulcloper

RT_003 **Julio 2008**

RNPS No **2142**
ISSN Solicitado

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Selección y construcción de objetos para el mejoramiento de un clasificador supervisado: un análisis crítico

Milton García Borroto, Yenny Villuendas Rey, Miguel Ángel Medina Pérez, Juliett Martínez López, José Ruiz Shulcloper.

Centro de Bioplantas y Facultad de Informática UNICA, Carretera a Morón km 9 ½, Ciego de Ávila, Cuba
Centro de Aplicaciones de Tecnología de Avanzada, 7a #21812 e/ 218 y 222, Siboney, Playa, Habana, Cuba
mil@bioplantas.cu

RT_003 CENATAV
Fecha del camera-ready: 3 de enero 2008

Resumen: El presente reporte técnico aborda la problemática del mejoramiento del desempeño y/o eficiencia de un clasificador supervisado desde la perspectiva de la selección y construcción de objetos. Se realiza un estudio crítico de los métodos más importantes reportados en la literatura. Cada método fue programado y se desarrolló una plataforma que permitió la realización de experimentos utilizando cada uno de ellos con diferentes bases de datos. Se ofrecen, además, los pseudocódigos y cuatro taxonomías que agrupan estos métodos, con el objetivo de hacer más fácil su estudio y aplicación.

Palabras clave: selección de objetos, construcción de prototipos, clasificación supervisada, vecino más cercano, vecino más similar.

Abstract: This technical report faces the improvement of supervised classifiers accuracy and/or efficiency by object selection and construction. A critical study of the most relevant methods reported is made. Each method was programmed and a framework which allows us to make experiments with different databases was developed. Also, pseudo codes and four taxonomies which group these methods are offered, to make easier its study and application.

Keyword: Object selection, prototype construction, supervised classification, nearest neighbor, most similar neighbor.

Contenido

Notación y acrónimos

U	Espacio de representación de los objetos pertenecientes a un universo U
$K_1, K_2, \ldots, K_l$	Clases en las que se estructura U
$R = \{r_1, r_2, \ldots, r_n\}$	Conjunto de rasgos en términos de los cuales se describen los objetos de U
$r_i(x)$	Valor del rasgo r_i en el objeto x
$\alpha(x)$	Índice de la clase a la que pertenece el objeto x, es decir, $x \in K_i$
$\Re$	Conjunto de los números reales.
$\lvert X \rvert$	Cardinal del conjunto X .
$\arg \underset{x}{Max}\{Rc_x\}$	Valor de x donde la relación o función Rc_x alcanza su máximo.
$\arg \underset{x}{Min}\{Rc_x\}$	Valor de x donde la relación o función Rc_x alcanza su mínimo.
$d(x, y)$	Distancia de x a y .
$D(x, y)$	Disimilaridad entre x y y .
$\mathrm{G}(X, \theta)$	Grafo G con un conjunto de nodos X y aristas $\theta \subset X \times X$.
$clasif(A, x)$	Resultado de la clasificación del objeto x utilizando el clasificador *clasif* entrenado con los objetos de A
$neighbor_k(x, A)$	Conjunto de los k vecinos más cercanos del objeto x en el conjunto A
$neighborSC_k(x, A)$	Conjunto de los k vecinos más cercanos x, de su misma clase, en el conjunto A
$NN(k, A, x)$	Resultado de la clasificación del objeto x utilizando el clasificador *k-NN* entrenado con los objetos de A
$clustering(A)$	Devuelve los centroides de los agrupamientos obtenidos por un clasificador no supervisado aplicado a A
First(A)	Devuelve el primer elemento del conjunto ordenado A
$Element_x$	Conjunto, valor numérico u objeto asociado a un objeto dado x
$mean(A)$	Centroide de A, calculado con la media aritmética por rasgo
$NUN(x, A)$	Vecino del objeto x más cercano de clase diferente (Nearest Unlike Neighbor) entre los elementos del conjunto A
$permute(A)$	Permuta de forma aleatoria los elementos de un conjunto totalmente ordenado A.
$sortAsc(A, \prec)$	Ordena ascendentemente los elementos del conjunto A de acuerdo a la relación de orden $\prec$.
$A[i]$	i-ésimo elemento de un conjunto totalmente ordenado A.
$B \overset{x}{\Leftarrow} A$	Mueve el elemento x de un conjunto A a un conjunto B

$\varepsilon_A^{clasif}(B)$	Número de errores que comete el clasificador *clasif*, entrenado con A, al clasificar los objetos de B.
$random(a,b)$	Función que genera un número aleatorio entre a y b
$randomPartition(A,ns)$	Genera una partición aleatoria de un conjunto A en ns subconjuntos.
$randomS(A,np)$	Devuelve un subconjunto aleatorio de A con np objetos.
$randomSC(A,K,np)$	Devuelve un subconjunto aleatorio de A con np objetos pertenecientes a la clase K.
$randomSCP(A,np)$	Devuelve un subconjunto aleatorio de A con np objetos. Este subconjunto contiene una cantidad de objetos por clases proporcional a la cantidad de objetos por clases de A.
BT4	Bootstrap 4
C&L	Clustering and Relabeling
CNN	Condensed Nearest Neighbor
CSE	Compact Set Editing
DSM	Decision Surface Mapping
EOP	Editing by Ordered Projection
GGE	Editing by Gabriel Graphs
IB2	Instance Based 2
LVQ-1	Learning Vector Quantization 1
LVQTC	Learning Vector Quantization with Training Counters
MCS	Minimal Consistent Subset
PNN	Prototypes for Nearest Neighbor
RNE	Editing by Relative Neighbors
RMHC	Random Mutation Hill Climbing
RNN	Reduced Nearest Neighbor
TIBL	Typical Instance-Based Learning
VQ	Vector Quantization
ENN	Edited Nearest Neighbor

1 Introducción

1.1 Selección de objetos de entrenamiento para la clasificación supervisada

La revolución computacional de los 90, principalmente en cuanto al abaratamiento de las memorias de alta capacidad y las altas velocidades de procesamiento de las computadoras personales, ha hecho resurgir el interés por modelos de algoritmos con elevado costo computacional, tanto algorítmico como espacial. Uno de los clasificadores más populares pertenecientes a esta categoría es la Regla del Vecino más Cercano (NN). Dado un conjunto T de objetos, de cada uno de los cuales se conoce su pertenencia a las clases del problema (matriz de entrenamiento), esta regla asigna a un objeto x que se desea clasificar la clase del objeto más cercano en el conjunto, de acuerdo a una medida de disimilaridad definida en el dominio del problema. Una extensión muy utilizada consiste en utilizar las clases de los k vecinos más cercanos (k-NN) para la determinación de la clase de x, utilizando algún procedimiento de integración, como la clase mayoritaria.

Además de las ventajas comunes a las técnicas de clasificación que no utilizan conocimiento previo sobre las distribuciones de los datos (o no paramétricas, como también se les conoce), la regla del vecino más cercano y su extensión a los k vecinos más cercanos combinan su simplicidad conceptual con el hecho de que el error asintótico es menor que el doble del error de Bayes[1]. El error de Bayes es el mínimo error teórico que puede cometer un clasificador, basado en las distribuciones estadísticas de sus datos [2]. Sin embargo, las reglas NN tienen algunas deficiencias importantes. En primer lugar, con matrices de entrenamiento con muchos objetos y atributos la clasificación se realiza de forma poco eficiente, ya que para clasificar un objeto su descripción tiene que ser comparada con las de todos los objetos de T. En segundo lugar, la matriz de entrenamiento puede contener ruido, u objetos con clases erróneamente asignadas, lo que deteriora la eficacia.

Muchas soluciones a ambos problemas han sido creadas y reportadas en trabajos previos, utilizando estrategias y algoritmos de variada naturaleza. Esto hace muy difícil no sólo una clasificación exhaustiva, sino una simple enumeración de todos los resultados alcanzados. No obstante, algunos trabajos han hecho comparaciones entre varios métodos con diferentes bases de datos, tanto reales como sintéticas [3, 4].

Los problemas de eficiencia y eficacia del NN se han abordado desde dos perspectivas. La primera es la *construcción* (o *generación*) *de objetos*, creando nuevos objetos no presentes necesariamente en el conjunto de entrenamiento, que contengan la información más relevante, y utilizando éstos para la clasificación final. La segunda metodología es la selección de un subconjunto de T, lo que se denomina *selección de objetos*. En esta metodología pueden distinguirse en la literatura dos familias [5, 6]. La primera de ellas la constituyen los algoritmos de *reducción* o *condensación*, que se centran en encontrar un subconjunto mínimo de objetos con los que se pueda obtener (aproximadamente) la misma eficacia que con la utilización de todo el conjunto original [7-10].

La segunda familia la constituyen los métodos de *edición basada en el error* (o simplemente *edición*), que tratan de eliminar los objetos ubicados en clases erróneas, así como minimizar las

zonas donde el error de Bayes es mayor. Estas técnicas suelen brindar mejoras en la eficacia de clasificación, usualmente con poca reducción del número de objetos [11-14]. Existen varias razones por las que un k-NN puede clasificar erróneamente un objeto x [15]:

1. Hay ruido presente en las cercanías de x en T. Los objetos ruidosos ganan el voto mayoritario, por lo que se predice la clase incorrecta.
2. El objeto x está cercano a las fronteras de decisión de las clases, donde la discriminación es más difícil, por la presencia de representantes de varias clases.
3. Los representantes de la clase correcta en la región de x son muy pocos, y los de otra clase más alejados ganan el voto mayoritario. Estos casos pueden presentarse en esquemas que consideran todos los vecinos con la misma importancia, independientemente de su disimilaridad con x, y utilizan valores de k grandes.
4. El problema no es eficazmente soluble con un k-NN, bien porque los datos están muy dispersos, o los atributos y/o la disimilaridad utilizada no reflejan la naturaleza del problema. Este es el caso, por ejemplo, de cuando los atributos escogidos no guardan relación con el problema de clasificación a resolver.

La mayoría de los métodos tratan de resolver los problemas 1 y 2, ya que el 3 se remedia con valores pequeños de k. Desafortunadamente no hay forma algorítmica alguna de diferenciar un objeto ruidoso (también conocido como *outlayer*), de otro que representa un conocimiento nuevo, diferente a lo común. En el caso del 4 posiblemente requiere cambiar toda la modelación del problema, lo que puede incluir seleccionar otros atributos, cambiar la función de analogía entre objetos o utilizar otro clasificador.

Aunque no existe una delimitación rigurosa entre las técnicas de edición y de condensación y se han mezclado en muchos estudios, es importante contrastar la naturaleza fuertemente heurística de los métodos de condensación contra el fuerte basamento estadístico de los métodos de edición más populares [16]. Sin embargo es importante resaltar que varias reglas asintóticamente óptimas, como por ejemplo el conocido Multiedit [17], pueden obtener resultados de baja calidad en problemas donde la cantidad de objetos no es suficientemente grande, en comparación con la dimensionalidad implícita del espacio de atributos. La dimensionalidad implícita de un problema está relacionada con la menor cantidad de atributos que pueden describirlo correctamente y es frecuentemente medida utilizando la covarianza entre los atributos [18]. Esto hace de la edición un problema más crítico que la condensación, lo que ha provocado varias mejoras y alternativas a los algoritmos tradicionales para problemas reales [19, 20].

Otra distinción entre editar y condensar se presenta por la diferencia, relativamente sutil, en el propósito. Debido a que la edición se utiliza para limpiar T de muestras erróneamente clasificadas, el objetivo fundamental es mejorar la eficacia de clasificación produciendo un conjunto "limpio". En este caso la obtención de un beneficio computacional tiene una importancia secundaria. La condensación, sin embargo, es utilizada primariamente con el propósito de reducir el número de muestras para ganar en eficiencia computacional, aunque esto se hace tratando de minimizar el cambio en la eficacia de clasificación. Desafortunadamente, este último objetivo muchas veces no se logra y estos métodos con frecuencia degradan apreciablemente la calidad del clasificador. Como resultado, la reducción debido a la edición es frecuentemente pequeña comparada con los métodos de condensación, pero se obtiene una eficacia de clasificación superior. Esta es la razón por la cual, mientras que la edición es preferida por el analista, la condensación es de mayor importancia práctica para el que desarrolla un sistema de reconocimiento de patrones con limitaciones de tiempo real [16].

La diferencia de propósito y de características entre la edición y la condensación sugieren una relación sinérgica, que ha sido explotada de forma implícita y explícita en varios trabajos [16, 21, 22]. Una sinergia es la integración de elementos que da como resultado algo mayor que la simple suma de sus componentes, por lo que, si dos elementos se combinan sinérgicamente, crean un resultado que aprovecha y maximiza las cualidades de cada uno de los elementos.

1.2 Datos mezclados e incompletos

Frecuentemente la representación de los objetos es considerada como una secuencia de valores exclusivamente numéricos o no numéricos. Sin embargo, existen muchos problemas reales de reconocimiento de patrones donde, en la descripción de cada objeto, ambos tipos de valores aparecen mezclados (ver, por ejemplo, las bases de datos del Repositorio de la UCI [23] y las aplicaciones en [24, 25]). Además, en muchas ocasiones existen objetos de los que el valor de uno o más de sus atributos es desconocido. A este tipo de problemas nos referiremos como problemas con Datos Mezclados e Incompletos (DMI) [26]. Existen 4 estrategias generales para lidiar con los datos mezclados, que se exponen a continuación.

1.2.1 Convertir los datos numéricos a no numéricos

Este proceso se realiza utilizando métodos de discretización (para una revisión consultar [27]). Éstos están basados frecuentemente en histogramas, cálculo de la entropía, o alguna técnica de construcción de árboles de decisión. En este caso decidir el número de categorías puede ser difícil y lo que es más importante, cuando las categorías son tratadas como nominales, la información del orden se pierde [28].

Por otro lado existen atributos que, desde el punto de vista de un especialista, no deben ser discretizados, ya que la similaridad entre ellos depende de la diferencia entre los valores y no del intervalo al que pertenezcan después de la discretización [29]. Por ejemplo, la similaridad entre los valores de muchas magnitudes físicas (campo magnético, anomalías aerogamma espectrométricas, intensidad de la corriente, etc.) dependen del error de medición, o de un umbral de similaridad que establece el especialista en función del problema en cuestión y no del intervalo en que ambos valores se encuentren.

1.2.2 Convertir los datos no numéricos a numéricos

Existen varias estrategias para realizar este proceso. Una de ellas, conocida como introducción de variables tontas (*dummy variables*) [30], convierte cada valor de la variable no numérica en una nueva variable Booleana, verdadera si el objeto tiene ese valor, o falsa si no lo tiene. Finalmente *verdadero* y *falso* son convertidos en 1 y 0 respectivamente.

En otros trabajos se transforman directamente cada valor no numérico en un código numérico, aplicándose después operaciones aritméticas sobre ellos. Este enfoque es cuestionable por dos razones fundamentales: la primera que si los valores originales no tienen un orden definido, el orden asociado a sus códigos puede no tener sentido alguno. Por ejemplo, para el atributo “localización del dolor al ingresar” se tienen los valores “no tiene”, “derecha”, “izquierda” e “indefinida”. Si los códigos 0, 1, 2 y 3 son asignados para cada valor, en el orden que aparecen, la distancia euclidiana de “no tiene” a “izquierda” es el doble de la de “no tiene” a

"derecha", lo que puede no ser razonable desde la perspectiva de un especialista [28], y la segunda que las operaciones aritméticas con estos códigos no tienen sentido, aun cuando exista un orden definido entre los valores no numéricos. Por ejemplo, asignándole 1, 2 y 3 a "bueno", "regular" y malo, ¿qué sentido tiene la expresión 3.1 * 1 + 1.26 * 2 (3.1 * "bueno" + 1.26 * "regular")?

Otra estrategia, introducida por Dutch et al. [31], utiliza una idea similar a la utilizada en la distancia VDM (Value Distance Metric) [32] para transformar cada atributo no numérico en un conjunto de atributos probabilísticos, uno por clase. De esta forma, al igual que en VDM, se substituye cada valor no numérico a por las probabilidades de que un objeto con ese valor pertenezca a cada clase.

Recientemente Maudes et al. [33] explotaron una idea similar a la utilización de un clasificador inicial para los datos nominales, siendo su salida para cada clase incluida como atributos numéricos del problema original. De esta forma se substituyen todos los atributos nominales por una t-upla de valores numéricos, uno por clase, que son las probabilidades asignadas por un clasificador de la pertenencia de ese conjunto de valores a cada clase del problema. En el artículo se comparan resultados utilizando un árbol de decisión (construido con C4.5) y una máquina de vectores soporte (SVM), con un núcleo (kernel) lineal.

Con respecto a estas dos últimas soluciones expuestas, hay que tener en cuenta que, aunque los nuevos valores introducidos son números, representan probabilidades, y no valores de atributos. Entonces, en el caso de datos mezclados, pueden aparecer inconsistencias cuando se comparan con los atributos numéricos presentes en la descripción de los objetos.

1.2.3 Aplicar un clasificador apropiado para cada tipo de atributos

Con esta estrategia se separan los datos numéricos de los no numéricos, y se aplica un clasificador diferente para cada tipo de dato. Finalmente los resultados de los dos clasificadores son utilizados para predecir la clase de un nuevo objeto. Esta idea permite utilizar todo el repositorio de clasificadores que ya existen para cada tipo de atributo. Esta estrategia tiene como desventaja fundamental que puede perder relaciones interesantes entre datos de diferente tipo, como, por ejemplo, que un valor de fiebre (atributo numérico) puede compararse de manera diferentes según el grado de dolor que tenga un paciente (atributo no numérico).

1.2.4 Trabajar directamente con los datos mezclados e incompletos

Las disimilaridades heterogéneas trabajan directamente con datos mezclados, utilizando diferentes criterios de comparación de atributos para los datos numéricos y los no numéricos. Existen varios resultados en esta dirección [34] y algunos estudios comparativos con distancias clásicas [28]. En este caso no se requieren transformaciones, por lo que no se pierde información. Sin embargo, varias de las funciones reportadas no cumplen con algunas de las propiedades de una distancia y hay que tener cuidado cuando se utilizan en un algoritmo que lo requiera.

Pekalska, Duin y otros autores han utilizado clasificadores basados en disimilaridades [35, 36]. Estos no utilizan objetos y funciones de disimilaridad entre ellos, sino que trabajan directamente sobre una matriz de comparaciones. Estas matrices pueden no tener las propiedades asociadas a las matrices construidas utilizando distancias. Hay que señalar que con esto no se

resuelve el problema, sino que se elude, partiendo de que ya el problema se resuelve en una etapa previa.

En el Reconocimiento Lógico-Combinatorio de Patrones (LCPR) [37, 38] existen variadas técnicas y algoritmos de eficacia probada en problemas prácticos [24, 25] para trabajar directamente con datos mezclados.

En cuanto al tratamiento de la ausencia de información existen muchos artículos y libros escritos al respecto [39]. Para una revisión del estado del arte consultar [40].

El trabajo está estructurado en las siguientes partes:

- **Introducción.**
- **Conceptos básicos.** Se presentan al lector los conceptos necesarios para la comprensión del documento.
- **Métodos de edición y de condensación.** En estas dos secciones se presentan varios métodos de edición basada en el error y de condensación, respectivamente. Los métodos escogidos muestran una cantidad considerable de ideas diferentes, sin olvidar los esquemas más populares reportados en la literatura especializada. De cada método se muestra lo siguiente:

  Introducción. Se introduce el método utilizando como referencia el artículo original donde apareció. Se describe brevemente la forma de operar, y se explica el por qué funciona.

  Algoritmo. Formalización del algoritmo, para eliminar los puntos oscuros y las confusiones que a veces se encuentran en la literatura. Es importante hacer notar que en algunos casos la descripción no se corresponde textualmente con la brindada por el autor en el artículo original, sino que ha sido rescrita de manera equivalente, pero más simple y clara. Los algoritmos han sido comentados paso a paso, para facilitar la comprensión.

  Valoración. Ventajas, desventajas y comentarios más importantes sobre el método, reportados en la literatura y nacidos de las evaluaciones teóricas y experimentales de los autores. Éstas son mostradas gráficamente en ejemplos con bases de datos sintéticas, creadas para destacar los aspectos más relevantes.

  Los métodos no serán presentados en orden cronológico por fines didácticos. Se mostrarán primero los métodos de selección de objetos, y dentro de éstos los de *edición* y luego los de *condensación*, en orden creciente de complejidad en cada caso. Finalmente, se mostrarán los métodos de construcción de objetos.
- **Taxonomía de los métodos de selección y construcción de objetos.** Varias taxonomías son explicadas, las que ayudan a entender los métodos y seleccionar el más apropiado para un problema en particular.
- **Clasificación taxonómica de los métodos de selección y construcción de objetos.** Se detalla la clasificación de cada método de acuerdo a las taxonomías enunciadas.
- **Resultados experimentales.** Se describen los resultados alcanzados con todos los métodos expuestos empleando bases de datos del repositorio de la UCI [23].
- **Referencias**
- **Anexos.** Se dan las descripciones de los resultados numéricos obtenidos en las aplicaciones de todos los métodos en todas las bases de datos empleadas en este trabajo.

1.3 Bases de datos sintéticas

Tres bases de datos sintéticas fueron utilizadas como ejemplos. Cada una de ellas tiene objetos pertenecientes a tres clases (representadas por círculos, triángulos y cuadrados) con su

correspondiente región de decisión (blanca, plateado y gris respectivamente). En las figuras los objetos originales se representan más pequeños, con el interior rellenado, mientras que los objetos resultantes del método son más grandes y tienen sólo el borde, como se muestra en la Fig. 1.

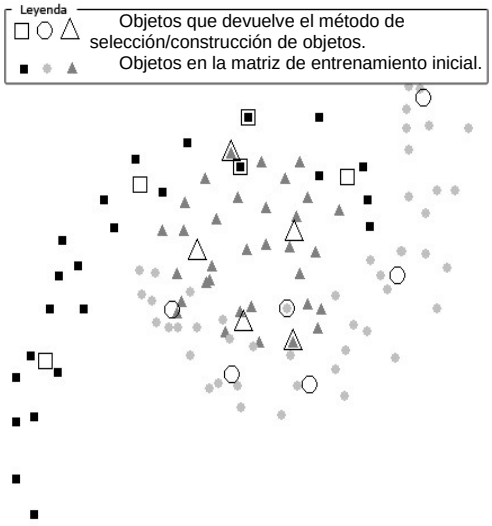

Fig. 1 Ejemplo de resultado de un método.

Los objetos de la primera base de datos se muestran en la Fig. 2, denominada "Banana y Pelota", que es una ligera variación de la que se ha utilizado reiteradamente de Bananas. Esta base de datos tiene 108 objetos y se utiliza para mostrar de forma general el resultado de los algoritmos. La distribución general de las clases no es trivial, y posee clases mezcladas y objetos fuera de la región que ocupan la mayoría de los objetos de su clase (outlyers).

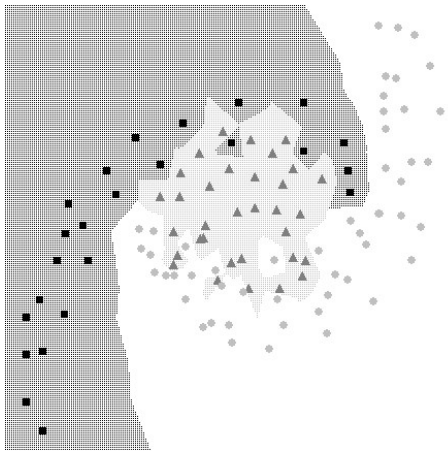

Fig. 2 Objetos originales de la base de datos "Banana y Pelota".

La segunda base de datos (Fig. 3) simula un diagrama de Venn con 3 clases altamente mezcladas, tiene 72 objetos y es utilizada para mostrar el funcionamiento de los métodos en estos casos.

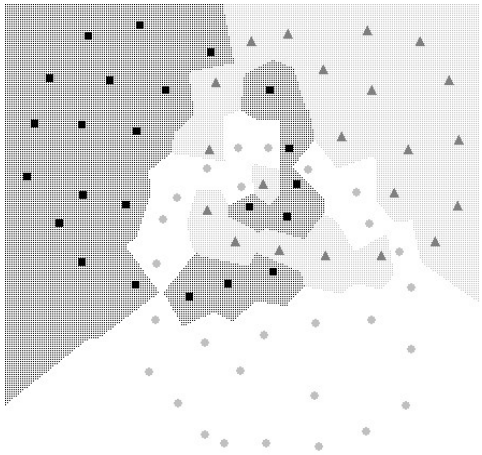

Fig. 3 Objetos originales de la base de datos "Diagrama de Venn". Se muestra un alto índice de mezcla entre las clases.

La última base de datos representa una situación ideal, con las clases perfectamente separadas (Fig. 4), y se utiliza fundamentalmente para mostrar el funcionamiento de los métodos en estas condiciones. Tiene un total de 185 objetos.

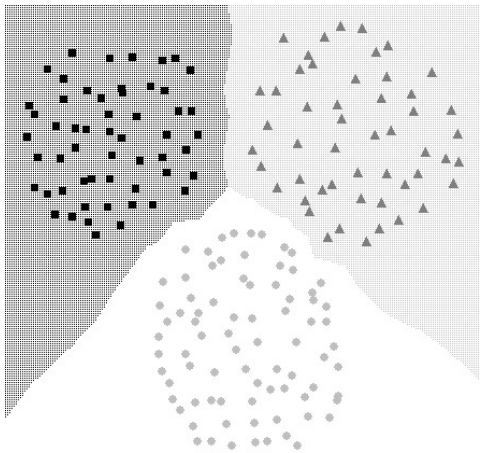

Fig. 4 Objetos originales de la base de datos "Bolas". Se muestran las clases totalmente separadas.

Es importante destacar que no todos los métodos tienen una muestra de su ejecución con todas las bases de datos, pues haría demasiado extenso el documento, sino que serán utilizados selectivamente en cada caso, para destacar los aspectos que se consideren más interesantes. De la misma forma, algunas bases de datos que se utilizan sólo en un método para visualizar algún comportamiento en particular, no fueron incluidas en este epígrafe.

El presente reporte dista mucho de estar completo, pues el tema de los métodos de selección y construcción de objetos está muy lejos de estar cerrado. Continúan apareciendo frecuentemente artículos con nuevas ideas, o mejoras de ideas anteriores. Además, cada método ha recibido a lo largo del tiempo una serie de mejoras, versiones y generalizaciones, de las cuales enunciamos aquí sólo algunas.

2 Conceptos básicos

2.1 Los objetos y su representación

El concepto de objeto es primario y por tanto muy difícil de definir en función de otros conceptos más simples. En este trabajo se considera como objeto una entidad entendible por el ser humano, que interviene en un problema de reconocimiento de patrones (RP). El conjunto de todos los objetos se llamará universo U. Ejemplos de objetos pueden ser embriones de zanahoria, pacientes, huellas dactilares, entre otros.

Usualmente los objetos están descritos por un conjunto de n atributos $R = \{r_1, r_2, \ldots, r_n\}$, que son extraídos del problema. Cada atributo está definido en un dominio $D_i = dom(r_i)$. La función $\begin{matrix} r_i : U \to D_i \\ x \to r_i(x) \end{matrix}$ asocia a cada objeto el valor $r_i(x)$ de su atributo r_i. El dominio de definición de un atributo es su conjunto de valores admisibles. En dependencia de este conjunto se pueden tener atributos de diferentes tipos, por ejemplo: Booleanos ($D_i = \{0,1\}$); *k*-valentes ($D_i = \{0,1,\ldots,k-1\}, k > 2$); Reales ($D_i \subseteq \Re$); etc.

Es importante destacar que usualmente no se trabaja directamente con los objetos, sino con su descripción en función de los atributos seleccionados. En este documento utilizaremos la palabra *objeto* para referirnos tanto a los objetos, como a sus descripciones. En cada caso se puede saber a cuál se hace referencia a partir del contexto.

El desconocimiento del valor de un atributo r_i en un objeto x se denota con el símbolo "?", el que se adiciona al conjunto de valores admisibles de r_i. Es decir, si $? \in D_i$ puede haber desconocimiento del valor del atributo $r_i(x)$ en algún objeto x.

2.2 Funciones de analogía entre objetos

Las funciones de analogía son un componente importante de muchos métodos estadísticos, de aprendizaje y de reconocimiento de patrones. Son ampliamente utilizadas tanto en problemas de

clasificación supervisada, como de agrupamiento y muchas veces determinan la calidad del resultado final.

La elección de la forma de comparar dos objetos debería ser extraída de la modelación del problema, según el concepto de similaridad que utiliza el especialista en el dominio. Sin embargo, en muchos trabajos donde se utilizan repositorios estándar de bases de datos, esto no es posible. En estos casos se escoge entre un amplio repertorio de funciones, que ya han sido utilizadas con anterioridad, o se crean las propias. Es importante destacar que la representación de los objetos impone restricciones importantes en cuanto a las funciones que se pueden aplicar. Es por esto que existen varios trabajos dedicados a la creación de funciones para diferentes dominios [34, 41], así como estudios comparativos entre sus características [28].

Es importante destacar que usualmente no se trabaja con los valores originales, sino que estos se normalizan primero. Esto se hace para que el resultado no sea afectado por el rango de definición de cada atributo. De no hacerse así, un atributo definido, por ejemplo, en el intervalo $[0,10000]$ puede hacer que su influencia anule totalmente a otro definido en $[10,20]$. Otro aspecto que se incorpora en muchos sistemas es el pesado de los atributos, para reflejar el hecho que no todos tienen la misma importancia para un problema dado.

Entre las funciones de analogía que admiten datos mezclados e incompletos, cabe señalar la HEOM (Heterogeneous Euclidean-Overlap Metric). Esta se define como:

$$HEOM(x,y)=\sqrt{\sum_{a=1}^{m} d_a(x_a,y_a)^2} \qquad (1)$$

donde d_a depende del tipo del atributo a, siendo:

$$d_a(x,y)=\begin{cases} 1 & \text{si el valor del atributo } a \text{ en } x \text{ o } y \text{ es desconocido} \\ overlap(x,y) & \text{si el atributo } a \text{ es nominal} \\ rn\_diff_a(x,y) & \text{en otro caso} \end{cases}$$

donde

$$overlap(x,y)=\begin{cases} 0, \text{ si } x=y \\ 1, \text{ en otro caso} \end{cases}$$

y

$$rn\_diff_a(x,y)=\frac{|x-y|}{\max_a-\min_a}$$

Posteriormente Wilson y Martínez [34], propusieron varias extensiones, como por ejemplo la HVDM (Heterogeneous Value Difference Metric), donde la función de analogía de valores de atributo d_a fue substituida por la utilizada en VDM [32]:

$$vdm_a(x,y) = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^q \quad (2)$$

donde:

- $N_{a,x}$: Cantidad de objetos del conjunto de entrenamiento que tiene el valor x en el atributo a,
- $N_{a,x,c}$: Cantidad de objetos de clase c del conjunto de entrenamiento que tiene el valor x en el atributo a
- C: Cantidad de clases
- q: es una constante, usualmente 1 o 2

Al utilizar vdm_a, dos valores son considerados como cercanos si tienen clasificaciones más similares (correlaciones más similares con el valor de clase), independientemente del posible orden de los valores.

También se ha utilizado VDM con datos mezclados, discretizando los valores numéricos [42]. Aunque discretizar tiene los inconvenientes ya señalados, puede dar resultados superiores en algunos problemas, por ejemplo, para seleccionar las personas que son buenos candidatos para pilotear un avión de combate específico. En este caso las personas con alturas significativamente mayores o menores al tamaño óptimo, deben ser consideradas malos candidatos. Por tanto, a los efectos de la clasificación, estos valores pueden considerarse como similares, aunque numéricamente sean muy diferentes [34].

Hay que señalar que, aunque sus autores les dan el nombre de "distancias" heterogéneas, en caso de existir ausencia de información, la función no cumple la propiedad de ser definida positiva, por lo que no es una distancia.

En este trabajo utilizaremos el término más general de disimilaridad para catalogar a las funciones de analogía entre objetos que devuelven valores más pequeños cuanto más similares son los objetos comparados. Utilizaremos el término de distancia para señalar el subconjunto de las disimilaridades que cumplen con las propiedades de: ser definida positiva, simetría y desigualdad triangular.

2.3 Criterios de consistencia

Sea un problema de clasificación supervisada con matriz de entrenamiento $T \subset U$. Se aplica un método de selección o construcción de objetos en *T* y se obtiene una nueva matriz *T'*. Históricamente se han identificado algunos criterios que parecen ser importante que posea el nuevo conjunto de objetos:

- Consistencia con respecto al conjunto de entrenamiento. *T'* es consistente con respecto a *T* si el clasificador entrenado con *T'* clasifica correctamente todos los objetos de *T*, o sea, $\forall x \in T[clasif(T,x) = clasif(T',x)]$. Usualmente es referido sólo con el término *consistente*.
- Consistencia con respecto a la frontera de decisión [43]. *T'* es consistente con respecto a *T* si determina exactamente la misma frontera de decisión, o sea, $\forall x \in U[clasif(T,x) = clasif(T',x)]$. Claramente esto implica la consistencia con respecto al conjunto de entrenamiento, pero el recíproco no es necesariamente cierto.

- Consistencia con respecto a las subclases [44]. Sea Φ una partición de T en subclases tales que $\forall i \in I \forall \Phi_i \in \Phi[(x_1, x_2 \in \Phi_i) \Rightarrow (\alpha(x_1) = \alpha(x_2))]$ y $X_i \subset \Phi_i$, un conjunto de representantes asociados a cada subclase. $X = \bigcup_{i \in I} X_i$ es consistente con respecto a las subclases $\Phi_i \in \Phi$ sí y sólo si: $\forall i \in I \; \forall x \in T \left[(x \in \Phi_i \Rightarrow k\text{-}NN(X, x, 1) \in \Phi_i \right]$

Estos criterios guían en muchos casos el proceso de búsqueda, pero no son universales, ya que muchos otros métodos generan resultados inconsistentes. En cada método de este trabajo se señalan las propiedades que satisfacen (ver epígrafe 6. Clasificación taxonómica de los métodos).

3 Métodos de edición

El ENN (Edited Nearest Neighbor) es el primer método de *edición basada en el error* reportado en la literatura. Fue creado por Wilson en 1972 [11] y consiste en la eliminación de todos los objetos que son mal clasificados por un k-NN, usualmente con k=3. Este proceso se realiza por lotes, por lo que los objetos son marcados primero y después se eliminan simultáneamente todos los marcados, lo que garantiza la independencia del orden. Ha sido utilizado extensivamente en las comparaciones experimentales con otros métodos, incluso en la actualidad.

En 1976 Tomek propone una variante, conocida como Allk-NN [12], que utiliza un procedimiento similar a ENN, pero con un conjunto de valores de k. Si un objeto es mal clasificado al menos por un k-NN, es eliminado del conjunto de entrenamiento. Su autor mostró una clara superioridad del método en comparación con ENN y la repetición continua de ENN mientras se puedan realizar eliminaciones.

Por su parte Devijver y Kittler [17] presentaron Multiedit en 1980. Inicialmente se particiona aleatoriamente el conjunto de entrenamiento en n componentes T_i. Posteriormente se aplica un ENN en cada componente T_i, siendo el clasificador que utiliza ENN entrenado con T_{i+1}. Finalmente para T_n se utiliza T_0. Después de eliminados los objetos se agrupan los resultantes nuevamente y se repite el proceso, hasta que un número definido de iteraciones no modifiquen el resultado. Aunque sus autores demostraron que para una muestra infinita se obtienen resultados óptimos, para matrices reales, con pocos objetos, puede incluso eliminar clases completas.

Un método utilizando grafos de proximidad (para una introducción a los grafos de proximidad, ver epígrafe anterior) fue introducido por Sánchez *et al*. en 1997 [20]. Este método consiste en aplicar la misma idea general que ENN, pero utilizando los vecinos en un grafo de proximidad. De esta forma se descartan todos los objetos que serían mal clasificados por un voto mayoritario de sus vecinos. Este método tiene varias ventajas sobre los esquemas anteriores [16]. En primer lugar el número de vecinos es variable y depende de la estructura interna de las clases del problema. Además, los vecinos de un objeto tienden a estar distribuidos alrededor de él, la información extraída de los objetos cercanos a las fronteras de decisión (donde la incertidumbre es mucho mayor) puede ser más relevante.

En el 2000 Hattori y Takahashi [45] publicaron un artículo con el criterio de que si al menos uno de los k vecinos más cercanos de un objeto era de clase contraria, éste era eliminado, proponiendo también un algoritmo para la determinación del valor óptimo de k. En el trabajo sólo se compararon contra ENN, y utilizando pocas bases de datos (la mayoría sintéticas). Esta

idea fue extendida en el 2003 por Sánchez *et al*. [13], incluyendo otros criterios de vecindad (centroide más cercano) y aplicándolo iterativamente.

Wang y Chen [14] propusieron en el 2005 la idea de utilizar para editar la noción de *colapso gravitacional*. Este es un concepto que se usa en astronomía para describir una nube de materia que tiene suficiente masa como para adquirir una fuerza gravitacional propia hacia su centro. En su método esto produce que las clases se contraigan espacialmente, por lo que se minimizan las zonas de mezcla. Para lograrlo modelaron la atracción gravitatoria con una ecuación que depende de la posición de los objetos y calculan la resultante de las fuerzas aplicadas sobre un objeto para determinar su dirección de movimiento. Cada objeto es movido de acuerdo a la fuerza que actúa sobre él, y las nuevas fuerzas son calculadas. El proceso es repetido un número determinado de pasos, obteniéndose el conjunto final. Nótese que este algoritmo no elimina ningún objeto. El método es independiente del orden, ya que el cálculo de las fuerzas se hace siempre con las posiciones de los objetos de la iteración anterior.

3.1 Edited Nearest Neighbors (ENN)

El Edited Nearest Neighbor (ENN) es el primer método de edición basada en el error reportado en la literatura. Fue creado por Wilson en 1972 [11] y consiste en la eliminación de todos los objetos que son mal clasificados por el k-NN. Este proceso se realiza por lotes, por lo que los objetos son marcados primero, y después se eliminan simultáneamente todos los marcados.

3.1.1 Algoritmo

Entradas:

T Matriz de entrenamiento
k Cantidad de vecinos a considerar en la asignación de la clase

Salida:

V Resultado del método

Pasos:

1	$Bad \leftarrow \phi$	El conjunto Bad contiene los objetos a eliminar
2	foreach $x \in T$ if $NN(k,T,x) \neq \alpha(x)$ then $Bad \leftarrow Bad \cup \{x\}$	Si el resultado de la clasificación de x difiere de su clase real, éste será eliminado
3	$V \leftarrow T \setminus Bad$	El resultado final está formado por los objetos bien clasificados
4	end	

3.1.2 Valoración

Ventajas: No depende del orden ni del azar, ya que utiliza los resultados de la aplicación de un clasificador. Trabaja bien con bases de datos pequeñas y con clases mezcladas [46].

Comentarios: Este método tiende a eliminar los objetos de la frontera y los aislados (Fig. 5), debido a que éstos son generalmente mal clasificados. Como trabaja con información local, puede dejar conjuntos de objetos totalmente aislados de su clase (Fig. 6).

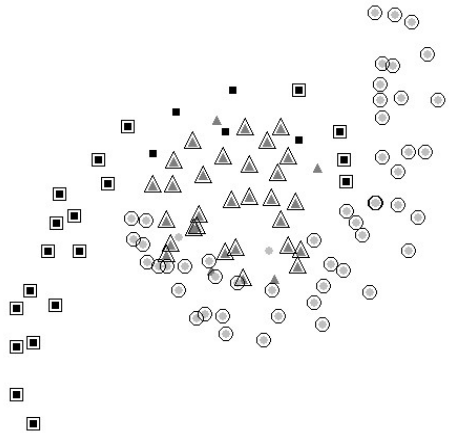

Fig. 5 ENN aplicado a "Banana y Pelota" con un 3-NN.

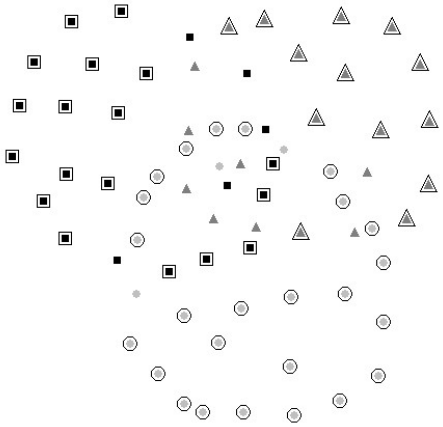

Fig. 6 ENN aplicado a "Diagrama de Venn" con un 3-NN.

3.2 Allk-NN

Este método, desarrollado por Tomek en 1976 [12], es una de las variantes del ENN de Wilson. Consiste en aplicar varias pasadas de k-NN con distintos valores de k, y eliminar todo objeto que haya sido mal clasificado en cualquiera de ellas.

3.2.1 Algoritmo

Entradas:

T Matriz de entrenamiento

$k_{\max}$ Cantidad máxima de vecinos a considerar en la asignación de la clase

Salida:

V Resultado del método

Pasos:

1	$Bad \leftarrow \phi$	El conjunto Bad contiene los objetos a eliminar
2	foreach $x \in T$	
	if $\exists k \in \{1,\ldots,k_{max}\}\left(NN(k,T,x) \neq \alpha(x)\right)$ then $Bad \leftarrow Bad \cup \{x\}$	El elemento x es eliminado si es mal clasificado para algún valor de k .
3	$V \leftarrow T \setminus Bad$	El resultado final está formado por los objetos bien clasificados
4	end	

3.2.2 Valoración

Ventajas: No depende del orden ni del azar. En el artículo original Tomek comparó este método con el ENN y con la repetición ilimitada del ENN (hasta que no hubieran modificaciones), encontrando una clara superioridad del All-kNN.

Desventajas: Por la heurística que utiliza, en problemas con clases muy mezcladas puede eliminar un número excesivo de objetos.

Comentarios: El autor también encontró evidencias que sustentan que la aplicación ilimitada del All-kNN puede obtener resultados con distribuciones de probabilidad no mezcladas. En general, tiende a quedarse con los objetos interiores de las clases, eliminando los fronterizos y los ruidosos. Su reducción va a depender de la separación entre las clases (Fig. 8).

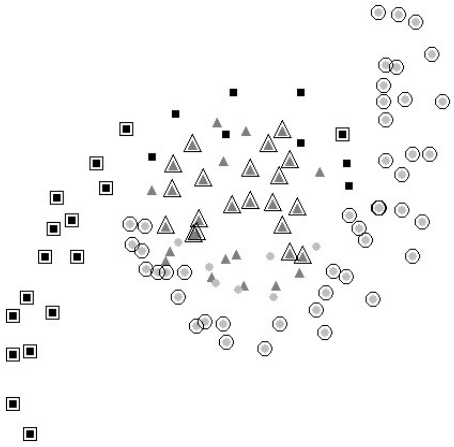

Fig. 7 Allk-NN (k=10) aplicado a "Banana y Pelota".

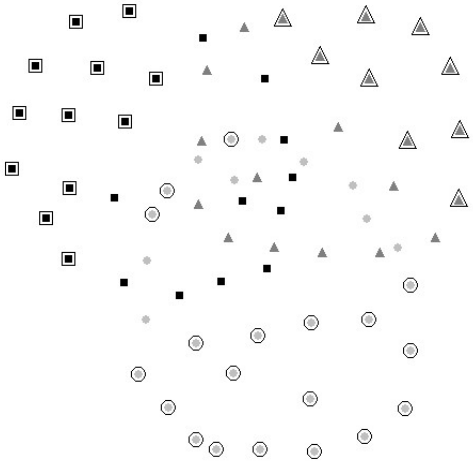

Fig. 8 Allk-NN (k=10) aplicado a "Diagrama de Venn".

3.3 Multiedit

Este método fue desarrollado por Devijver y Kittler en 1980 [17]. Consiste en la partición aleatoria de la matriz de entrenamiento en *ns* componentes, aplicando después a cada partición un ENN que utiliza un clasificador 1-NN entrenado con la partición siguiente. Después de cada iteración, se unen las particiones resultantes y se repite el proceso hasta que no existan cambios en *ni* iteraciones sucesivas.

3.3.1 Algoritmo

Entradas:

T Matriz de entrenamiento

ni Número de iteraciones máximas que se ejecutarán sin cambios en el resultado del algoritmo

ns Número de subconjuntos al particionar, $ns \geq 3$

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow T$	El resultado inicial es toda la matriz de entrenamiento.
2	$it \leftarrow 0$	it controla la cantidad de iteraciones sin cambios en el conjunto resultante
3	$\{V_i\} \leftarrow randomPartition(V, ns)$	El resultado actual se particiona aleatoriamente en ns subconjuntos
4	$Bad \leftarrow \phi$	
5	Foreach $i \in \{1,\ldots,ns\}$ $Bad \leftarrow \{x \in V_i : NN(1, V_{(i+1) \bmod ns}, x) \neq \alpha(x)\}$	Cada objeto es clasificado con un 1-NN entrenado con la siguiente partición. Si es mal clasificado, será eliminado
6	if $\lvert Bad \rvert = 0$ then $it \leftarrow it + 1$ else $V \leftarrow V \setminus Bad$ $it \leftarrow 0$	Si no se han encontrado nuevos objetos mal clasificados, se incrementa it, si no éstos son eliminados, y se resetea el contador it a 0
7	if $(it < n_i)$ then goto 3	
8	end	

3.3.2 Valoración

Ventajas: La homogeneidad de los agrupamientos resultantes brinda la estructura ideal para procesamientos posteriores, pues brinda una gran separabilidad en las regiones de las clases. Según Kuncheva [47] trabaja bien en grandes conjuntos de datos con clases en forma de nubes.

Desventajas: Para bases de datos pequeñas, o con clases mezcladas, puede eliminar todos los objetos de una clase [47] (Fig. 9b). Si las clases están perfectamente separadas, prácticamente no elimina objetos. Depende del azar y del orden de presentación de los objetos, pues se entrena con diferentes particiones que son aleatorias.

Comentarios: Produce agrupamientos homogéneos, es decir, en un agrupamiento sólo hay muestras de una clase. Cada agrupamiento expande la región de decisión de Bayes asociada a la clase correspondiente [17]. Al comienzo de cada iteración, el conjunto se considera representativo de las distribuciones subyacentes. Cada subconjunto es considerado como una muestra aleatoria de estas distribuciones.

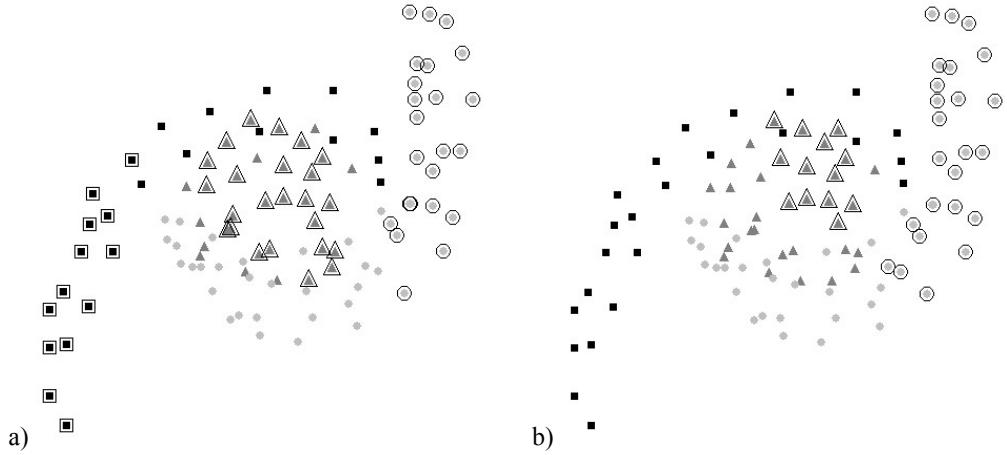

Fig. 9 Dos aplicaciones del método Multiedit (ns=3, ni=5) a "Banana y Pelota". a) Separación total de las clases. b) Eliminación total de una clase.

4 Métodos de condensación

El objetivo de los métodos de condensación es reducir significativamente la cantidad de objetos en la matriz de entrenamiento, con la menor afectación posible a la eficacia del clasificador resultante. Existen dos estrategias fundamentales: seleccionar o construir objetos [3].

En la selección de objetos para condensar se busca un subconjunto de la matriz de entrenamiento original, tan pequeño como sea posible, que contenga la información más relevante del conjunto original. En la construcción de objetos, por otra parte, se crean nuevos objetos, no necesariamente pertenecientes a la matriz original. Es importante en este caso resaltar que los nuevos objetos pertenecen al dominio de representación de los atributos, pero no necesariamente al dominio de objetos del problema.

4.1 Métodos de selección de objetos

El primer método de condensación fue creado por Hart [7] y fue denominado CNN (por sus siglas en inglés: Condensed Nearest Neighbor). Se basa en el concepto de subconjunto consistente, que es un subconjunto del conjunto de entrenamiento (T), que es suficiente para clasificar todo T utilizando 1-NN. Se basa en la construcción, paso a paso, de un nuevo conjunto de objetos V, moviendo hacia él cada elemento de la matriz de entrenamiento T, si éste es erróneamente clasificado por los objetos que ya están en V. El proceso se repite hasta que se eliminen todos los errores (inconsistencias).

Este método logra altos niveles de compactación (eliminación de objetos) en matrices en que el riesgo de Bayes no es muy alto, pues si no, la compactación puede ser ínfima. Es altamente dependiente del orden de presentación de los datos, por lo que frecuentemente se comienza por permutar los objetos al azar. Por la forma que funciona tiende a quedarse con los objetos cercanos a la frontera de decisión de cada clase, ya que estos son fácilmente mal clasificados, y al ser incluidos evitan los errores en objetos interiores.

Como su mismo autor reconoce, usualmente este método no obtiene el subconjunto mínimo consistente, pudiendo quedar (en dependencia del orden de los objetos) objetos muy alejados de la frontera. Por ejemplo, el primer objeto siempre es adicionado al resultado.

Para tratar de eliminar las desventajas de CNN, diferentes variantes han sido propuestas:

- Gates en 1972 introduce RNN [9]. Es un postprocesamiento de CNN, para eliminar los objetos que no son necesarios para garantizar la consistencia. Éste consiste en tratar de eliminar cada objeto del resultado devuelto por el CNN, si con esto no se crea ninguna inconsistencia. Según su creador, si el subconjunto mínimo consistente se encuentra en el resultado de CNN, RNN lo encuentra. Sin embargo, en muchas ocasiones, el tiempo extra de búsqueda no justifica el nivel de compactación alcanzado, pues el costo computacional es muy superior al de CNN [10].
- Ullman en 1974 presenta dos variantes más [48]. La primera consiste en la adición de un umbral de "zona muerta" δ en la evaluación de la distancia mínima durante el proceso de identificar el conjunto condensado. Un objeto es eliminado solamente si la diferencia entre las distancias al objeto más cercano de clase diferente y de su clase es mayor que δ. La segunda introduce un ordenamiento de los objetos antes de ser evaluados, según su distancia a los elementos del resto de las clases. Esto elimina la dependencia del orden, al mismo tiempo que produce resultados con menos objetos.
- Tomek en 1976 [49] propone dos modificaciones, con el objetivo directo de retener los objetos de la frontera, identificando los vecinos más cercanos de otras clases
- En 1991 Aha propuso el IB-2 (Instance Based), inicialmente conocido como Growth Additive[50], que consiste en una sola iteración del método de Hart (CNN). Éste, aunque más rápido, es aún más dependiente del orden de los objetos, ya que cada iteración de CNN puede incluir objetos, que debido a la adición de otros al resultado en iteraciones anteriores, ahora estén mal clasificados. En general no devuelve un subconjunto consistente, y frecuentemente genera resultados muy poco representativos del conjunto original.
- Angiulli en 2005 [51] propuso FCNN (Fast CNN), una variante para eliminar la dependencia del orden, y acelerar la convergencia. Para ello utiliza la desigualdad triangular y las celdas de Voronoi, para eliminar la mayor parte de las comparaciones del algoritmo original. Todo esto permite su aplicación en bases de datos mucho mayores que su antecesor (en [51] se experimenta con bases de datos de hasta un millón de objetos, con dos atributos).

En 1972 Swonger [52] presentó ICA (Iterative Condensation Algorithm). Este método, a diferencia de CNN, permite la adición y eliminación de objetos del resultado. Otras de las ventajas de este método son la tolerancia a objetos ruidosos, la posibilidad de trabajar con objetos idénticos, pero de clases diferentes y la convergencia del esquema. Esto permite detener el proceso manualmente, antes de su condición de parada determinada.

Ritter propuso en 1975 [53] otro método para encontrar un subconjunto con tres características: consistencia, cercanía mayor a los objetos de su clase que los demás, y minimalidad. Este método, nombrado SNN (Selective Nearest Neighbor), es más complejo que sus antecesores, por la necesidad de cumplir las tres propiedades simultáneamente [10]. Además, como su mismo autor reconoce, ya que el criterio de SNN es más específico que el de CNN, el subconjunto resultante no es en general mínimo.

Un paso importante para la obtención del subconjunto mínimo consistente fue dado por Dasarathy en 1994 [10], al enunciar el algoritmo MCS (Minimal Consistent Subset). Aunque su autor consideró que la evidencia experimental aportada señalaba (aunque sin una demostración rigurosa) que se había alcanzado el objetivo, posteriormente varios autores han encontrado

contraejemplos [8, 54]. No obstante, este método ha sido ampliamente utilizado en comparaciones experimentales, por encontrar subconjuntos muy pequeños de manera determinística.

MCS está basado en el concepto de NUN [55] (Nearest Unlike Neighbor), que es el vecino más cercano, de clase diferente. Cada objeto x es asociado a los objetos y que cumplen que x es más cercano que el NUN de y. Estas asociaciones tienen la propiedad que, utilizando un clasificador 1-NN, si el objeto x permanece en el conjunto resultante, entonces y puede ser eliminado manteniéndose la consistencia. Por este motivo, en algunas referencias esto se denomina x *soporta* a y. Los objetos son ordenados descendentemente según la cantidad de objetos que soportan, y el primero es elegido. Luego se eliminan los objetos que ya son soportados por éste y se repite el paso anterior, hasta que ya no puedan eliminarse más objetos. En este conjunto resultante, varios objetos ya fueron eliminados, por lo que los NUN posiblemente cambiaron y se repite todo el proceso nuevamente con los nuevos objetos.

El problema general de la obtención del subconjunto mínimo consistente ha generado muchos intentos de solución, aunque Wilfong [56] demostró que es un problema *np*-completo.

También los grafos de proximidad [57] han sido utilizados para condensar [43]. Estos son grafos dirigidos donde cada objeto se conecta mediante un arco con sus vecinos, según cierto criterio de vecindad. Los tres grafos más utilizados para condensar son:

- Grafo de Voronoi (utilizado en [58]). Es el dual de una triangulación de Delaunay[59]. Cada objeto se conecta con otro si comparten una arista común de la triangulación. Es muy costoso de calcular, por lo que su uso se ha limitado a problemas muy pequeños, con pocas variables. Recientemente Takekazu y Toshikazu [60] crearon una forma más eficiente de calcularlo, y lo aplicaron a la selección de objetos, con resultados superiores al Quick Hull [61], el algoritmo más eficiente para calcular este grafo en espacios *n*-dimensionales.
- Grafo de vecinos relativos (utilizado en [20, 58]). En este grafo dos objetos x y y son vecinos si $\forall z \in T\left(d\left(x,y\right) < \max\left\{d\left(x,z\right), d\left(y,z\right)\right\}\right)$, o sea no existen otros objetos en la intersección de las hiperesferas centradas en los dos objetos, y de radio igual a la distancia entre ellas.
- Grafo de Gabriel (utilizado en [20, 58]). Dos objetos x y y son vecinos si $\forall z \in T\left(d\left(\frac{x+y}{2}, z\right) > \frac{d\left(x,y\right)}{2}\right)$, lo que significa que no existen objetos en la hiperesfera que tiene como centro el punto medio entre x y y, y de radio la distancia entre ambos.

La forma de utilizarlos consiste en eliminar todos los objetos que no tienen ningún vecino de clase diferente, por lo que se pueden considerar *interiores* a sus respectivas clases. En general tiene buen desempeño, quedándose con los objetos de la frontera, aunque han sido mostrados casos patológicos, en los que aún con clases totalmente separadas ningún objeto puede ser eliminado [57].

Muchos autores han reportado métodos con un fuerte componente aleatorio. El más simple, la búsqueda al azar [62], selecciona p subconjuntos aleatorios de cardinalidad fija n, y selecciona de éstos el que menor error obtenga al clasificar la matriz T. Ambos parámetros son definidos por el usuario y de ellos depende, en buena medida, la calidad del resultado. Los algoritmos genéticos también han sido aplicados, con diferentes estrategias y operadores [8, 19, 63], dando

buenos resultados, al igual que el ascenso de colina con mutaciones aleatorias [64]. Este tipo de estrategias han sido evaluadas y comparadas en varios artículos [3, 8, 65]. Los resultados encontrados sugieren que estos esquemas son competitivos con los demás sólo para problemas relativamente pequeños, pero pueden encontrar buenas soluciones en problemas con muchos objetos, donde los demás métodos son inaplicables por su alto costo computacional.

Un método especialmente diseñado para seleccionar las instancias típicas, denominado TIBL (Typical Instance Based Learning), fue propuesto por Zhang en 1992 [66]. La tipicidad de los objetos es calculada como el cociente de la distancia promedio de cada objeto a los objetos de diferentes clases y la distancia promedio a los objetos de la misma clase, asumiendo que los objetos más "típicos" están rodeados de otros objetos de su clase, mientras que están alejados de objetos de clase contraria. El conjunto resultante se construye mediante la repetición de los siguientes pasos: selección del objeto x más típico de los mal clasificados y adición al resultado del objeto más típico que causa que x se clasifique bien. Este algoritmo requiere modificaciones para manejar regiones geométricas disjuntas que pertenezcan a la misma clase, ya que el cálculo de la tipicidad toma en cuenta todos los objetos de la misma clase. Además, presupone que todos los objetos se comparan con una distancia fijada por el algoritmo, obviando el hecho de que la función de analogía puede ser dada por el experto en el área del problema.

TIBL puede lograr una reducción importante del número de objetos si las clases no están muy entremezcladas, obteniéndose un conjunto consistente, aunque puede tener dificultades para manejar problemas con fronteras de decisión entre clases complejas [67]. No existe una única forma de definir el entremezclado de las clases en una base de datos. Por el momento, utilizaremos la idea intuitiva que dos clases están entremezcladas si muchos objetos de una clase tienen objetos muy similares de la otra clase (más adelante expondremos una forma de cómo medirlo numéricamente). Por otra parte, la frontera de decisión teórica entre dos clases está formada por aquellos objetos en los que la probabilidad *a posteriori* de pertenecer a cada clase es la misma. En la práctica, consideramos a un objeto como perteneciente a la frontera, si dicha probabilidad es aproximadamente la misma.

En 2002 Zhang y Sun [54] desarrollaron un método basado en búsqueda tabú, un método meta-heurístico ampliamente utilizado para resolver problemas combinatorios [68]. La búsqueda tabú puede resumirse de la siguiente forma: comenzar con una solución inicial s (llamada configuración), y evaluarla con la función a optimizar. Construir posteriormente el conjunto de movimientos candidatos (configuraciones vecinas) $N(s)$, que son nuevas soluciones que se pueden construir a partir de s. Una de estas es elegida, y declarada tabú por un número predefinido de movimientos (tamaño de la lista tabú, o tabu tenure, en inglés). De esta manera la mejor solución es siempre almacenada y es la solución final del método. Para aplicarlo a la selección de objetos cada configuración es una cadena binaria, donde cada bit representa la inclusión o no del objeto en esa posición en el resultado final, y los movimientos candidatos se generan excluyendo o incluyendo a un objeto, según si está o no respectivamente en la configuración actual. Según sus autores, este método encuentra una aproximación mejor que las reportadas anteriores al subconjunto mínimo consistente.

Nuevos métodos son reportados constantemente en la literatura, como RE y WRE [69], basado en el cálculo de la *entropía representativa*, que está relacionada con la probabilidad de que un objeto quede representado en el conjunto resultante por un objeto.

En 2005 García-Borroto y Ruiz-Shulcloper reportaron CSE [44] (Compact Set Editing), un algoritmo basado en conjuntos β-compactos [70]. En este trabajo se introduce el concepto de consistencia con respecto a las subclases, que es un refinamiento del concepto de consistencia de

Hart, donde se agrega la propiedad que todo objeto tiene que tener un representante de su propia subclase en el conjunto resultante más cercano que cualquier representante de otra subclase. Esto asegura que el conjunto resultante sea representativo del original, mostrado en una amplia comparación experimental. La estructuración en subclases se realiza mediante la búsqueda de los conjuntos compactos [70](de ahí el nombre del método). Un conjunto compacto es un subgrafo conexo de un grafo de máxima similaridad.

El CSE se basa en la construcción del grafo de máxima similaridad (cada objeto está conectado con su o sus vecinos más similares, según la similaridad utilizada). Los grados de cada vértice (cantidad de aristas entrantes y salientes) son utilizados para elegir el objeto menos importante, para su eliminación. Al ser eliminado se garantiza la consistencia del resultado mediante un procedimiento de marcado de los objetos que le aseguran. Si un objeto es el último con una marca en particular, entonces es adicionado al resultado.

También las máquinas de vectores soporte (SVM) han sido utilizadas para esta tarea [71]. La ventaja de las SVM es que son construidas de acuerdo al principio de minimización del riesgo estructural, por lo que tienen buen poder de generalización, especialmente para bases de datos pequeñas. De esta forma, un subconjunto de los vectores de soporte encontrados al generar un SVM, pueden ser utilizados como matriz de entrenamiento de un clasificador NN.

Huang en el 2006 [72] creó un algoritmo utilizando la teoría de las relaciones estructurales grises (*Gray Relational Structures*, en inglés) [73]. Para esto se calculan dos coeficientes, el GRC (gray relational coeficient) y el GRG (gray relational grade), los que se utilizan para comparar cuán similar es un objeto de otro, basado en el valor de sus atributos. Con esta información se construye un grafo dirigido, donde cada objeto se enlaza con sus más similares. La información de los grados de entrada y salida de cada vértice, determina la permanencia o no del objeto relacionado en el conjunto resultante.

Fayed *et al*. [74] crearon SGP (Self Generating Prototypes) en 2007, basado en la idea de formar grupos de patrones de la misma clase. Inicialmente los patrones de cada clase forman un grupo único. Entonces se aplican un conjunto de operaciones simples, basadas en ciertos criterios a cumplir, por ejemplo dividir un grupo en varios, mover objetos de un grupo a otro y mezclar dos grupos diferentes. Finalmente el resultado está formado por el centroide de cada agrupamiento.

4.1.1 Random Search

Este método consiste en la selección del mejor resultado (en términos de eficacia) entre *ni* iteraciones de una selección aleatoria de *np* objetos de la matriz de entrenamiento. No obstante su sencillez, se han reportado buenos resultados en su utilización [8, 64]. También se le conoce como simulación de Monte Carlo.

4.1.1.1 Algoritmo

Entradas:

T	Matriz de entrenamiento
np	Número de objetos
ni	Número de iteraciones

Salida:

V	Resultado del método

Pasos:

1	$V \leftarrow randomS(T, np)$	Inicialmente el resultado es un conjunto aleatorio
2	repeat *ni - 1* times $NewV \leftarrow randomS(T, np)$ if $\varepsilon_{NewV}^{1NN}(T) < \varepsilon_{V}^{1NN}(T)$ then $V \leftarrow NewV$	Se genera un nuevo conjunto aleatorio Si el clasificador entrenado con el nuevo conjunto tiene menos errores, este se convierte en el resultado actual
3	end	

4.1.1.2 Valoración

Ventajas: Cuando el número de objetos es pequeño (en nuestros experimentos, menos de 200) obtiene conjuntos muy buenos (Fig. 10a).

Desventajas: Depende totalmente del azar (Fig. 10), pudiendo obtenerse en dos ejecuciones sucesivas un resultado muy bueno, y otro muy malo.

Comentarios: Los resultados experimentales sugieren que es un método muy efectivo en problemas con reducido número de objetos, sobrepasando incluso al resto de los métodos que se analizan. Cuando el número de objetos aumenta las combinaciones posibles crecen exponencialmente, por lo que para encontrar una combinación razonablemente buena, hay que elevar excesivamente el número de iteraciones, perdiendo el método su utilidad.

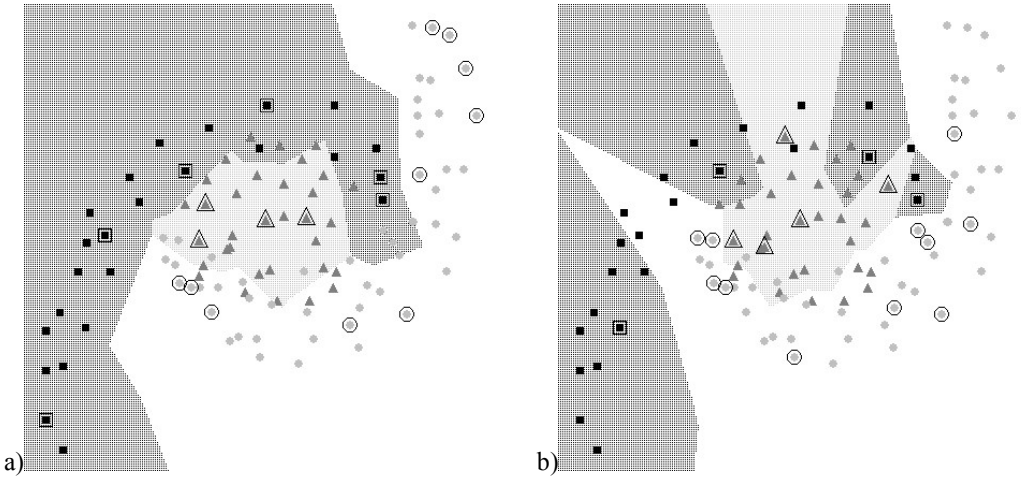

Fig. 10 Dos aplicaciones del Random Search (np=20, ni=100), a "Banana y Pelota".

4.1.2 Random Mutation Hill Climbing (RMHC)

Este método es un método evolutivo descrito originalmente por Mitchell y colaboradores en 1994 [65]. Inicialmente se selecciona una cadena binaria aleatoria (*Best*), y se evalúa su eficacia. La cadena es mutada en un bit aleatorio, obteniendo la nueva cadena (*New*), a la que se le calcula también la eficacia. Si la cadena mutada tiene una eficacia mayor que *Best*, entonces la

reemplaza. Este procedimiento es repetido un número seleccionado de iteraciones. Está diseñado para un clasificador 1-NN.

4.1.2.1 Algoritmo

Entradas:

T	Matriz de entrenamiento
ni	Número de iteraciones

Salida:

V	Resultado del método

Pasos:

1	foreach $i \in \{1,\ldots,T\}$ if $random(0,1)=1$ then $BinString[i] \leftarrow true$ then $BinString[i] \leftarrow false$	La variable $BinString$ contiene una cadena binaria, inicializada aleatoriamente.
2	$V \leftarrow buildSubset(T, BinString)$	La función $buildSubset(T, BinString)$ construye un subconjunto de T, adicionando el objeto $T[i]$ si $BinString[i]=true$
3	repeat ni -1 times $i \leftarrow random(0,\lvert T\rvert -1)$ $BinString[i] \leftarrow not(BinString[i])$ $NewV \leftarrow buildSubset(T, BinString)$ if $\varepsilon_{NewV}^{1NN}(T) < \varepsilon_{V}^{1NN}(T)$ then $V \leftarrow NewV$ else $BinString[i] \leftarrow not(BinString[i])$	Se selecciona un índice aleatorio. Se muta el elemento *i* Se construye el conjunto asociado a la nueva cadena binaria Si el clasificador k-NN comete menos errores con la nueva matriz, esta se convierte en el resultado actual, si no se restaura la cadena binaria a la versión anterior
4	end	

4.1.2.2 Valoración

Ventajas: La simpleza de la heurística utilizada da muy buenos resultados en bases de datos pequeñas.

Desventajas: Al igual que todos los esquemas aleatorios, depende fuertemente del azar (Fig. 11). Debido a su estrategia de búsqueda es muy propenso a caer en extremos locales, por lo que en muchos casos obtiene un número excesivo de objetos (Fig. 11a).

Comentarios: Al igual que el Random Search, funciona eficazmente con un número pequeño de objetos (ver comentario al respecto en la página 28). En [65] puede encontrarse un estudio comparativo preliminar entre varios métodos aleatorios y evolutivos.

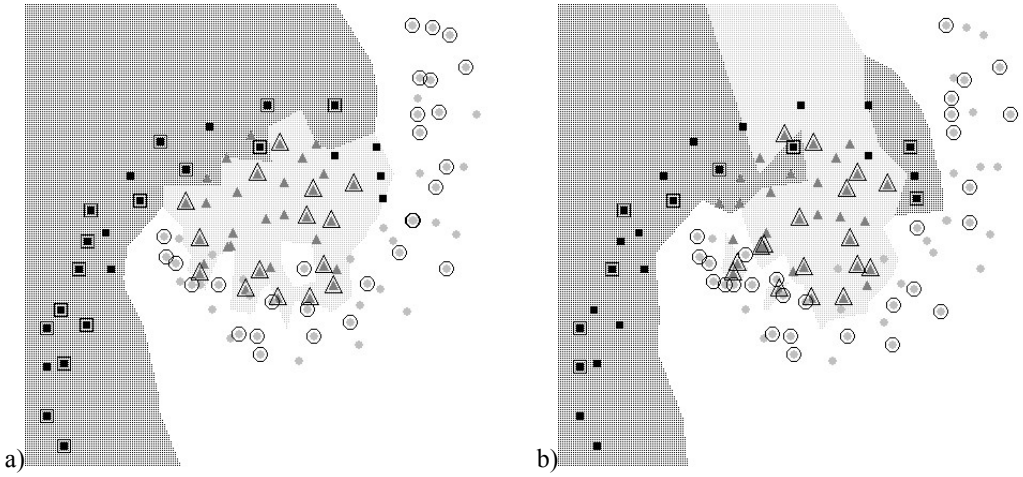

Fig. 11 Dos aplicaciones del RMHC (ni=100) a "Banana y Pelota".

4.1.3 Instance Based Learning (IB–2)

Fue presentado por Aha en 1991 [41], como miembro de una serie de algoritmos basados en instancias. El conjunto resultante *V* es construido adicionando los objetos de la matriz de entrenamiento que son mal clasificados con respecto a los elementos que ya están en *V*. Fue llamado inicialmente Growth (Additive) Algorithm, en un estudio preliminar [50]. Funcionalmente consiste en una sola pasada del método de Hart.

4.1.3.1 Algoritmo

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow \phi$	
2	foreach $x \in T$	
	if $NN(1,V,x) \neq \alpha(x)$ then $V \leftarrow V \bigcup \{x\}$	Si un clasificador 1-NN, entrenado con V, clasifica incorrectamente a x, éste es agregado al resultado
3	end	

4.1.3.2 Valoración

Desventajas: Depende críticamente del orden de presentación de los objetos, pudiendo obtenerse resultados muy poco deseables (Fig. 12).

Comentarios: Este algoritmo tiene mucho en común con el CNN de Hart, pero realiza una sola iteración, lo que genera que el resultado usualmente no sea consistente. Tiende a retener los

objetos de los bordes, ya que éstos son más propensos a ser mal clasificados, y elimina los objetos internos (Fig. 12b). Los objetos ruidosos, que son generalmente mal clasificados por encontrarse rodeados de objetos de clases diferentes, tienden a quedarse en el conjunto resultante.

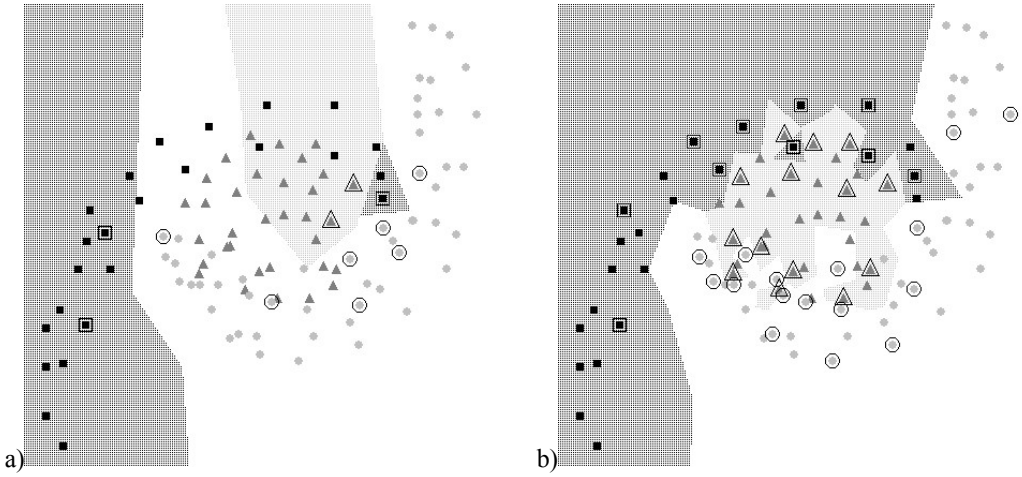

Fig. 12 Dos resultados de la aplicación del IB-2 (k=1) a "Banana y Pelota", permutando aleatoriamente los objetos. Nótese la crítica dependencia del orden.

4.1.4 Shrink

Este algoritmo, presentado por Kibler y Aha en 1987 [50], trabaja de manera contrapuesta al IB-2, eliminando de la matriz de entrenamiento los elementos que, al ser eliminados, siguen estando bien clasificados por el resto de los objetos que permanecen en la matriz.

4.1.4.1 Algoritmo

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow T$	Inicialmente el resultado es toda la matriz de entrenamiento
2	foreach $v \in V$ if $NN(1, V \setminus \{v\}, v) = \alpha(v)$ then $V \leftarrow V \setminus \{v\}$	Se elimina el objeto *v* si, al eliminarlo del resultado, se sigue clasificando correctamente
3	end	

4.1.4.2 Valoración

Desventajas: Al igual que el IB-2, es sensible al ruido. Es un método de una sola pasada. Depende del orden (Fig. 13), aunque no tan críticamente como el IB-2.

Comentarios: Tiende a eliminar los puntos interiores, ya que éstos están rodeados de vecinos de su misma clase, que les aseguran su buena clasificación, mientras retiene los cercanos a la frontera de decisión (Fig. 14).

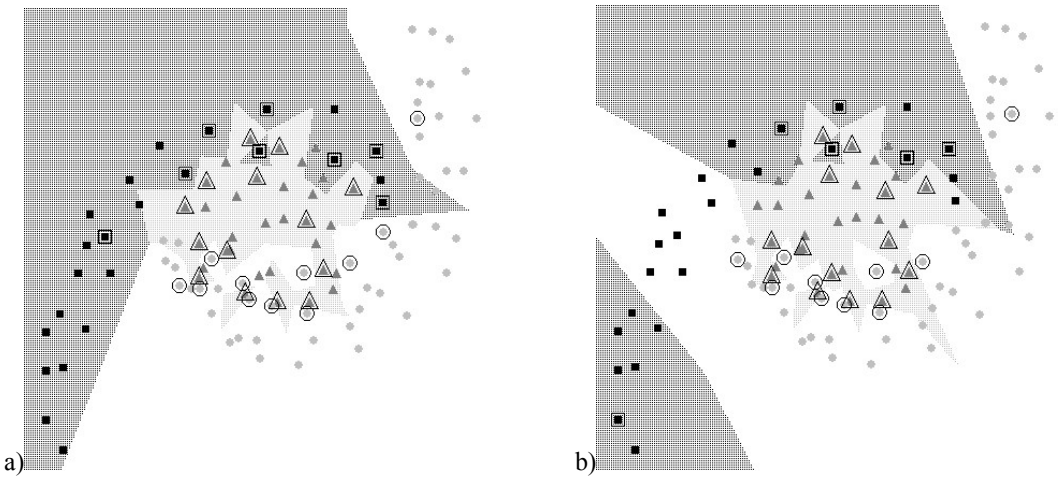

Fig. 13 Dos aplicaciones del Shrink (k=1) a "Banana y Pelota".

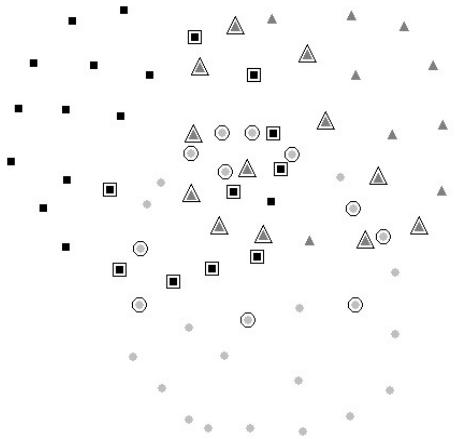

Fig. 14 Shrink aplicado a "Diagrama de Venn".

4.1.5 Condensed Nearest Neighbors (CNN)

El Condensed Nearest Neighbors (CNN) es el primer método de selección de objetos reportado en la literatura. Fue desarrollado por Hart en 1966 [7], y se basa en la construcción, paso a paso,

de un nuevo conjunto de objetos *V*, moviendo hacia él cada elemento de la matriz de entrenamiento *T*, si éste es erróneamente clasificado por los objetos que ya están en *V*.

4.1.5.1 Algoritmo

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow \phi$	
2	$GrabBag \leftarrow T$	Del conjunto $GrabBag$, inicializado con la matriz de entrenamiento, se moverán los elementos seleccionados al resultado
3	$x \leftarrow First(GrabBag)$ $V \overset{x}{\Leftarrow} GrabBag$	Se mueve el primer elemento
4	$moved \leftarrow false$	La bandera $moved$ señala si se realizó algún cambio en el resultado, en la iteración actual
5	foreach $y \in GrabBag$ if $NN(1,V,y) \neq \alpha(y)$ then $V \overset{y}{\Leftarrow} GrabBag$ $moved \leftarrow true$	Si el objeto y es mal clasificado con un 1-NN entrenado con el resultado actual, entonces y es movido a V, y la bandera $moved$ es activada
6	if $(GrabBag \neq \phi) \wedge moved$ then $goto$ 4	Si G*rabBag* no se encuentra aún vacío y se ha realizado algún cambio, comenzar una nueva iteración
7	end	

4.1.5.2 Valoración

Ventajas: Experimentos realizados por los autores con bases de datos artificiales muestran que logra un alto nivel de compactación en matrices con clases no mezcladas.

Desventajas: Es un método altamente dependiente del orden de presentación de los objetos (Fig. 15). Su propio creador plantea [7] que si el riesgo de Bayes es alto, entonces el conjunto resultante contendrá esencialmente todos los puntos presentes en el conjunto original y no se llevará a cabo ninguna reducción importante en el tamaño del conjunto (por ejemplo, en la Fig. 16, en la zona más solapada no ocurre casi reducción del número de objetos).

Comentarios: Tiende a quedarse con los puntos cercanos a la frontera de decisión de cada clase, ya que estos son los más propensos a ser mal clasificados, aunque dependiendo del orden, deja muchas veces muchos más de los necesitados para lograr la consistencia (Fig. 17).

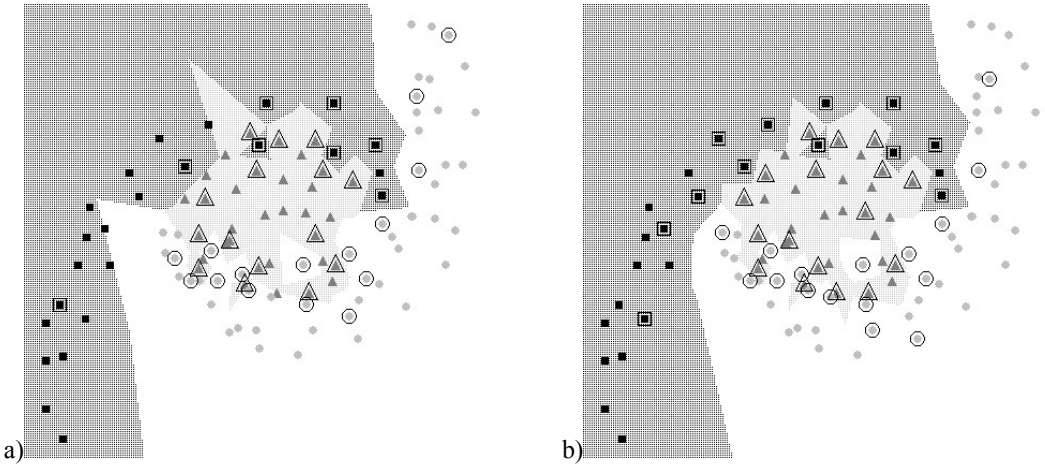

Fig. 15 Dos aplicaciones del CNN a "Banana y Pelota", permutando los objetos. Obsérvese las variaciones en la región de decisión y en el número de objetos (38 objetos en a) y 45 en b), aunque el resultado es siempre consistente).

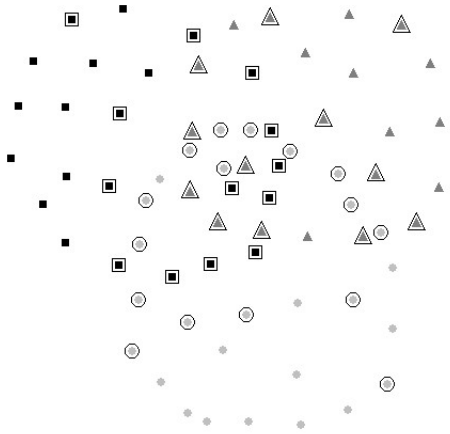

Fig. 16 CNN con "Diagrama de Venn".

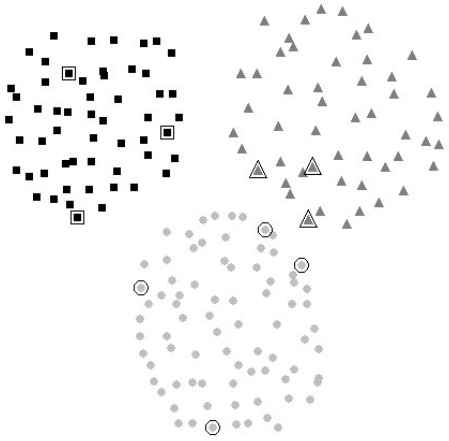

Fig. 17 CNN con "Bolas".

4.1.6 Reduced Nearest Neighbor (RNN)

Este método, creado por Gates en 1972 [9], trata de eliminar los objetos innecesarios (desde el punto de vista de garantizar la consistencia del resultado) en el conjunto resultante de la aplicación del método CNN. Estos son detectados ya que, al ser eliminados, no se introducen nuevos errores en el clasificador.

4.1.6.1 Algoritmo

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow CNN(T)$	El resultado inicial es obtenido mediante la aplicación del método CNN, también descrito en este trabajo
2	foreach $v \in V$ if $\varepsilon_{V \setminus \{v\}}^{1NN}(T) = 0$ then $V \leftarrow V \setminus \{v\}$	El objeto v se elimina si al hacerlo no se introduce ningún error en un clasificador k-NN entrenado con $V \setminus \{v\}$, clasificando todos los objetos de T
3	end	

4.1.6.2 Valoración

Ventajas: Se obtiene niveles de compactación superior al CNN. Según su autor, si la solución mínima consistente está incluida en el resultado del CNN, el RNN la encuentra [9].

Desventajas: Es dependiente del orden (ver Fig. 18). Según su propio autor [9], en muchos casos el tiempo extra de búsqueda de objetos para su eliminación no se justifica con el nivel de compactación extra alcanzado.

Comentarios: Obtiene un conjunto consistente, subconjunto del resultado del CNN. Al igual que éste tiende a eliminar los puntos interiores y a retener los bordes.

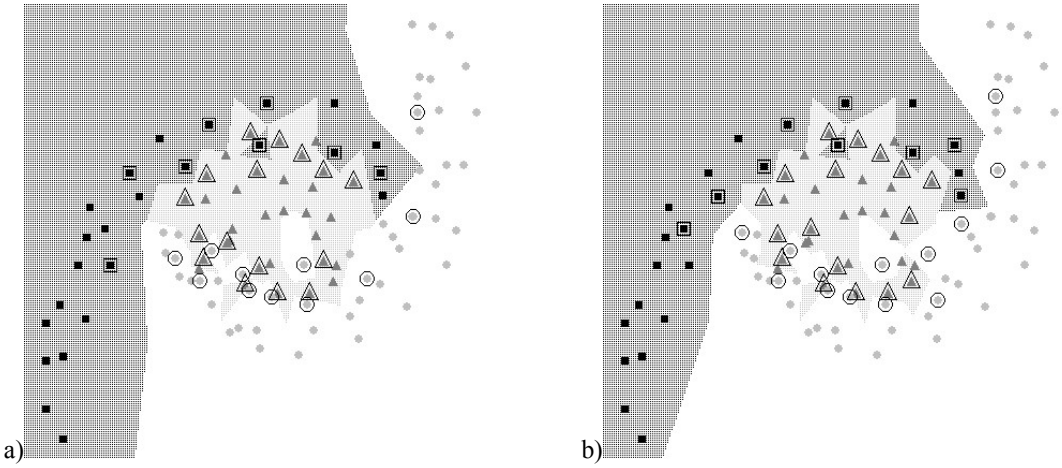

Fig. 18 Dos resultados de la aplicación del RNN a "Banana y Pelota", permutando aleatoriamente los objetos. Nótese que pueden obtenerse conjuntos de diferente tamaño, 35 objetos en a) y 39 en b).

4.1.7 Proximity Graphs Condensing (PGC)

Los grafos de proximidad han sido aplicados a varios problemas dentro del reconocimiento de patrones [43], dentro de los que se incluye la selección de prototipos. En este trabajo expondremos el algoritmo más utilizado para la condensación, el que puede ser aplicado con varios grafos de proximidad diferentes (el grafo de Vecinos Relativos y el grafo de Gabriel). El algoritmo consiste en retener los objetos con al menos un vecino de clase diferentes.

4.1.7.1 Algoritmo

Notación y definiciones:

Sea el grafo $G(X,\theta)$, se definen como vecinos de un nodo x a: $GNeighbor(x,G(X,\theta))=\{y\in X:(x,y)\in\theta\vee(y,x)\in\theta\}$

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$(U,+,\cdot)$	Solo está definida para el método buildGraph del Grafo de Gabriel
2	$G(T,\theta)\leftarrow buildGraph(T)$	Se construye el grafo de proximidad seleccionado
3	$V\leftarrow\{x\in T:\exists y\in GNeighbor(x,G(T,\theta))(\alpha(x)\neq\alpha(y))\}$	
4	end	

El método $buildGraph$ se implementa según el grafo de proximidad seleccionado

- Grafo de Gabriel: $G(X,\theta)$ es un grafo de Gabriel si y sólo si cumple que: $\forall x,y,z\in X\left((x,y)\in\theta\Leftrightarrow d\left(\frac{x+y}{2},z\right)>\frac{d(x,y)}{2}\right)$
- Grafo de Vecinos Relativos (VR): $G(X,\theta)$ es un grafo de vecinos relativos (RN) si y sólo si: $\forall x,y,z\in X\left((x,y)\in\theta\Leftrightarrow d(x,y)<\max\{d(x,z),d(y,z)\}\right)$

4.1.7.2 Valoración

Ventajas: Según [58] con estos métodos se logra una reducción importante del número de objetos, sin un deterioro apreciable de la eficacia del clasificador.

Desventajas: En casos no convencionales y clases muy mezcladas, puede producir reducciones ínfimas del número de objetos (Fig. 20).

Comentarios: Estos métodos permiten la utilización de cualquier grafo de proximidad, como el Grafo de Gabriel, Vecinos Relativos y diagrama de Voronoi. Este último grafo garantiza los

mejores resultados, obteniéndose un resultado que no modifica la frontera de decisión del clasificador, aunque el algoritmo de construcción del grafo de Voronoi es exponencial.

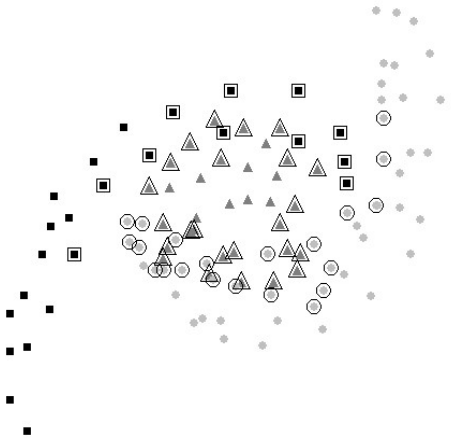

Fig. 19 Aplicación de PGC-Gabriel a "Banana y Pelota".

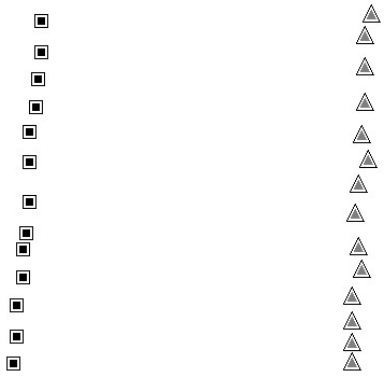

Fig. 20 Caso patológico del uso de PGC-Gabriel. Note que todos los objetos son seleccionados.

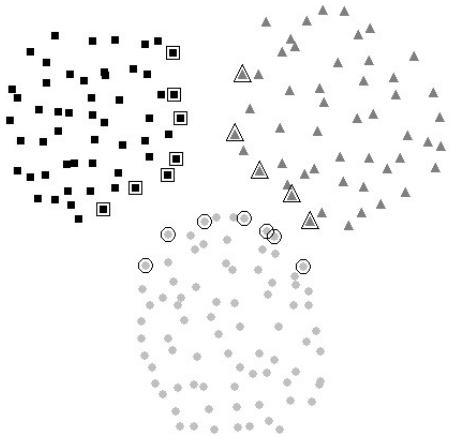

Fig. 21 Aplicación de GPC-Gabriel a "Bolas".

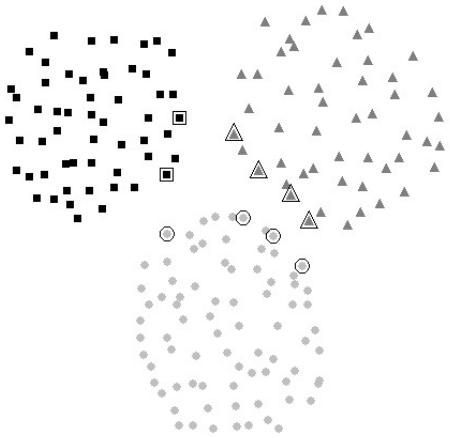

Fig. 22 Aplicación de GPC-Vecinos Relativos a "Bolas".

4.1.8 Editing by Ordered Projection (EOP)

Este método, desarrollado por Aguilar y colaboradores en el 2001 [75], consiste en la eliminación de los objetos alejados de la frontera de decisión del clasificador. A diferencia de otros métodos este proceso no es realizado utilizando una distancia entre objetos ni un clasificador específico, sino por la comparación de valores para cada rasgo. La siguiente heurística es utilizada: un objeto *x* está en la frontera si existe un rasgo *r* cuyo valor en algún objeto de clase contraria a *x* está más cercano que los valores de *r* de los objetos de su misma clase. Como puede apreciarse lo que se utiliza para la decisión es la proyección de las

descripciones de los objetos utilizando cada uno de los rasgos independientemente, de ahí el nombre del método.

4.1.8.1 Algoritmo

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow []$	
2	foreach $r_i \in R$ foreach $x \in T \setminus V$ $RightN \leftarrow \{y \in T : r_i(x) \leq r_i(y)\}$ $RMin \leftarrow \{y \in T : r_i(y) = \min_{z \in RightN} \{r_i(z)\}\}$ $LeftN \leftarrow \{y \in T : r_i(x) \geq r_i(y)\}$ $LMax \leftarrow \{y \in T : r_i(y) = \max_{z \in LeftN} \{r_i(z)\}\}$ if $\exists y \in RMin(\alpha(x) \neq \alpha(y)) \vee \exists z \in LMax(\alpha(x) \neq \alpha(z))$ then $V \leftarrow V \bigcup \{x\}$, Si al menos existe un rasgo para el que x tenga un vecino de otra clase se añade *x* al resultado, pues no puede ser eliminado	Para cada rasgo y para cada elemento de la matriz de entrenamiento que no esté en el conjunto resultante Objetos cuyo valor en el rasgo sobrepasa al valor del rasgo en el objeto *x* Vecinos derechos de *x* Objetos cuyo valor en el rasgo es menor que el valor del rasgo en el objeto *x* Vecinos izquierdos de *x*
3	end	

4.1.8.2 Valoración

Ventajas: Es el más rápido de todos los métodos discutidos en este trabajo, ya que no realiza ningún tipo de comparación entre los objetos, ni utiliza ningún clasificador. Según sus autores [75], presenta una considerable reducción en el número de ejemplos, aunque esto es muy dependiente de la distribución espacial de los datos.

Desventajas: Puede no editar mucho aún en clases totalmente separadas, ya que necesita también la separación de las proyecciones (Fig. 23). Presupone la existencia de un orden en los valores de los rasgos.

Comentarios: La eficacia de este método depende no sólo de la distribución de los datos, sino de la distribución de las proyecciones de sus rasgos. Es muy inestable a la rotación de las descripciones de los objetos con respecto a un centro.

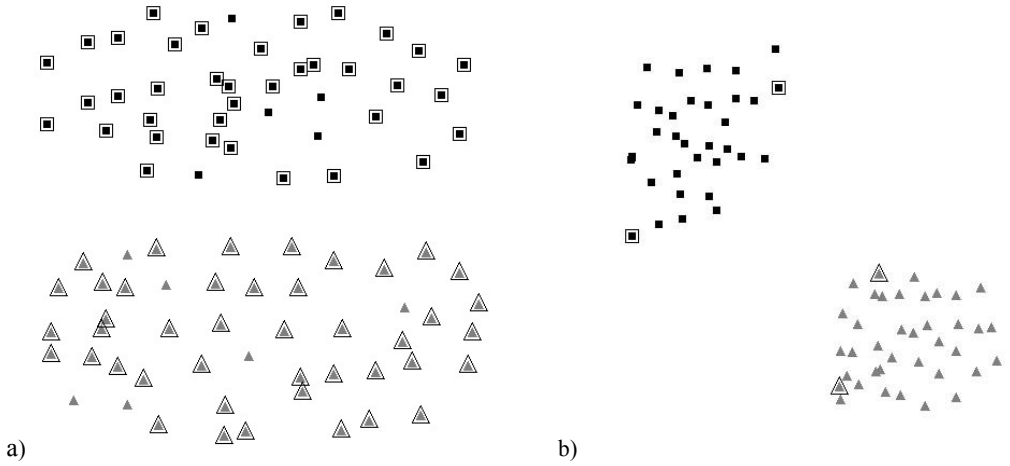

Fig. 23 Aplicación del EOP a) Caso patológico, poca condensación aun con clases totalmente separadas, ya que las proyecciones se mezclan. b) Mejor caso, las clases tienen proyecciones separadas.

4.1.9 Decremental Reduction Optimization Procedures (DROP – 3)

La familia de algoritmos DROP fue desarrollada por Randall Wilson en 1997 [67]. En este trabajo se analiza el DROP-3 (ENN + DROP-2), por considerarlo el más interesante. Todos los DROPs conciben el proceso de clasificación utilizando un k-NN.

4.1.9.1 Algoritmo

Notación y Definiciones:

$$x \prec y \Leftrightarrow d\left(x, NUN\left(x,T\right)\right) > d\left(y, NUN\left(y,T\right)\right)$$

Relación de orden utilizada en el algoritmo. Un elemento antecede a otro si su vecino más cercano de clase contraria (NUN) está más lejano, por lo que debe estar más alejado de la frontera de decisión.

Entradas:

T	Matriz de entrenamiento
K	Cantidad de vecinos a utilizar en el algoritmo

Salida:

V	Resultado del método

Pasos:

1	$NewT \leftarrow ENN\left(T\right)$	Se considera como resultado inicial el conjunto resultante de aplicar el método de edición ENN a T
2	$V \leftarrow NewT$	

3	foreach $x \in NewT$ $Associated_x \leftarrow \phi$	Para cada elemento, se inicializan sus asociados
4	foreach $x \in NewT$ $NN_x \leftarrow neighbors_{k+1}(x, NewT)$ foreach $y \in NN_x$ $Associated_y \leftarrow Associated_y \cup \{x\}$	$k+1$ vecinos más cercanos de x Para cada *y* vecino de x, x es entonces, asociado de *y*
5	$sortAsc(V, \prec)$	
6	foreach $v \in V$ $with \leftarrow \varepsilon_V^{kNN}(Associated_v)$ $without \leftarrow \varepsilon_{V\setminus\{v\}}^{kNN}(Associated_v)$ if $with > without$ then $V \leftarrow V \setminus \{v\}$ foreach $x \in (Associated_v \setminus \{v\})$ $NN_x \leftarrow neighbors_{k+1}(x, V)$ foreach $y \in NN_x$ $Associated_y \leftarrow Associated_y \cup x$	Error al clasificar los asociados, eliminando a v de la matriz de entrenamiento del *k-NN* Si el error con su presencia es mayor que sin ella, se elimina v del resultado. Para cada elemento asociado con él, se hallan sus $k+1$ vecinos más cercanos y se actualizan sus asociados
7	end	

4.1.9.2 Valoración

Ventajas: Logra altos índices de compactación (Fig. 24 y Fig. 25)

Desventajas: La aplicación del ENN y el algoritmo guiado puramente por la no disminución de la eficacia puede obtener conjuntos de objetos que no sean representativos del conjunto original (Fig. 24).

Comentarios: Este método tiende a eliminar primero los objetos internos, ya que se ordenan de acuerdo a la distancia a su vecino más cercano de clase contraria. La aplicación inicial de un ENN garantiza que los objetos ruidosos son eliminados. Según su autor [67] este método disminuye las posibilidades de sobreentrenamiento del clasificador.

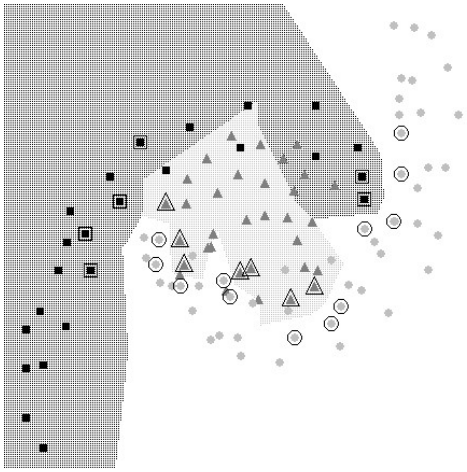

Fig. 24 Aplicación del DROP-3 al "Banana y Pelota".

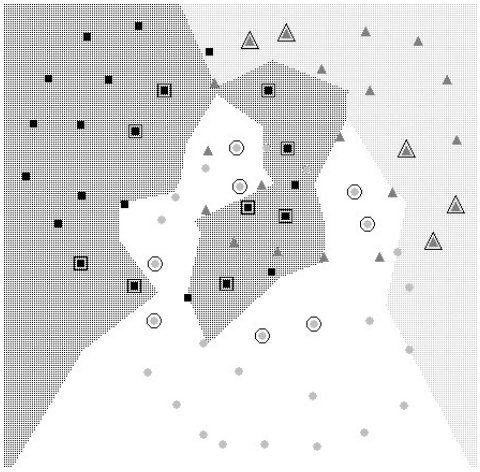

Fig. 25 Aplicación del DROP-3 a "Diagrama de Venn".

4.1.10 Typical Instance – Based Learning (TIBL)

Este algoritmo fue desarrollado por Zhang en 1992 [66]. Se basa en el concepto de tipicidad de los objetos, que en este caso es calculado como el cociente de la distancia promedio de cada objeto a los de diferentes clases y la distancia promedio a los objetos de la misma clase, asumiendo que los objetos más "típicos" están rodeados de otros objetos de su clase, mientras que están alejados de objetos de clase contraria.

El algoritmo consta de los siguientes pasos: Se toma el objeto más típico x incorrectamente clasificado con el conjunto V actual, adicionándose a V el objeto más típico que causa que x sea correctamente clasificado. Este proceso se repite hasta que se obtenga un conjunto consistente.

4.1.10.1 Algoritmo

Notación y Definiciones:

$$d(x,y) = \sqrt{\frac{1}{n}\sum_{r_i \in R}\left(\frac{r_i(x) - r_i(y)}{\max\limits_{z \in T}\{r_i(z)\} - \min\limits_{z \in T}\{r_i(z)\}}\right)^2}$$

Distancia utilizada en el algoritmo

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	foreach $x \; in \; T$ $Cx \leftarrow \{y \in T : \alpha(y) = \alpha(x)\}$ $Cnx \leftarrow \{y \in T : \alpha(y) \neq \alpha(x)\}$ $typicality_x \leftarrow \frac{\frac{1}{\lvert Cnx \rvert}\sum_{y \in Cnx} d(x,y)}{\frac{1}{\lvert Cx \rvert}\sum_{y \in Cx} d(x,y)}$	Para cada elemento de la matriz de entrenamiento, se buscan las instancias pertenecientes a su clase y las que no pertenecen a su clase y se calcula la tipicidad de cada objeto x
2	$V \leftarrow \phi$	
3	$Bc \leftarrow \{x \in T : NN(1,V,x) \neq \alpha(x)\}$	Se buscan las instancias mal clasificadas con el conjunto actual
4	$mt \leftarrow \arg\max_{y \in Bc} \{typicality_y\}$	De ellas, se selecciona la de mayor tipicidad
5	$dN \leftarrow d(mt, NUN(mt,V))$	Se calcula la distancia al NUN de la instancia más típica
6	$X \leftarrow \{x \in T : d(x,mt) < dN\}$	Se seleccionan las instancias de la misma clase más cercanas que el NUN
7	$z \leftarrow \arg\max_{y \in X} \{typicality_y\}$	Se busca la más típica de su clase
8	$V \leftarrow V \cup \{z\}$	Y se añade *z* a *V*
9	if $\varepsilon_V^{1NN}(T) \neq 0$ then goto 3	Si existen errores de clasificación se repite el proceso
10	end	

4.1.10.2 Valoración

Ventajas: Puede lograr una reducción importante del número de objetos, si las clases no están muy mezcladas (Fig. 26). Se obtiene un conjunto consistente.

Desventajas: Puede tener dificultades en problemas con frontera de decisión complejas[67] (Fig. 27). Requiere modificaciones para manejar regiones geométricas disjuntas que pertenezcan a la misma clase, ya que el cálculo de la tipicidad toma en cuenta todos los objetos de la misma clase. Presupone que todos los objetos se comparan con una distancia fijada por el algoritmo, obviando el hecho de que la función de analogía puede ser extraída de la modelación del problema.

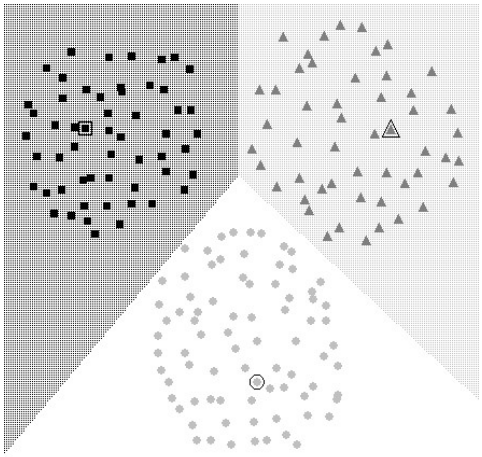

Fig. 26 Aplicación del TIBL a "Bolas".

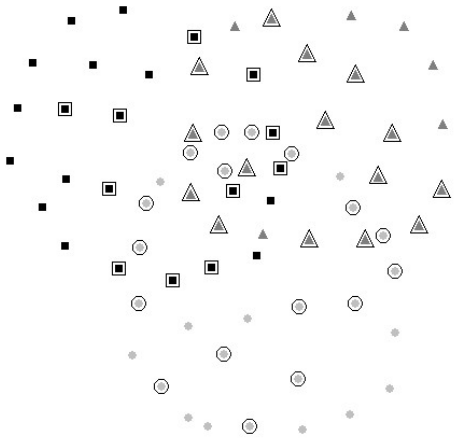

Fig. 27 TIBL aplicado a "Diagrama de Venn".

4.1.11 Minimal Consistent Subset (MCS)

Este método, propuesto por Dasarathy en 1994 [10], está basado en el concepto de Nearest Unlike Neighbor (NUN), vecino más cercano de clase diferente. Se entiende por *soporte de x elemento de la matriz de entrenamiento T* al conjunto de elementos de *T* que dista de *x* en menos de *dNun*, distancia de *x* a su NUN. Este método consiste en seleccionar aquellos elementos que aseguren la correcta clasificación de la mayor cantidad de objetos de su clase. Originalmente se concibe la utilización del clasificador 1-NN, aunque Dasarathy en [10] plantea que puede ser extendido para un k-NN.

4.1.11.1 Algoritmo

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$Current \leftarrow T$	El conjunto consistente inicial es toda la matriz de entrenamiento
2	$V \leftarrow \phi$	
3	foreach $x \in T$ $Voters_x \leftarrow \phi$	Para cada muestra de la matriz de entrenamiento inicializar su conjunto de votantes, conformado por las muestras de su soporte
4	foreach $x \in T$	Para cada muestra, hallar la distancia al NUN y las

	$dN \leftarrow d(x, NUN(x, Current))$ $Nearest_x \leftarrow \{y \in Current : d(y,x) < dN\}$ foreach $y \in Nearest_x$ $add(Voters_y, x)$ if $Nearest_x = \{x\}$ then $V \leftarrow V \bigcup \{x\}$	muestras de su soporte. Para cada una de las muestras que se encontraban más cerca que el NUN se actualiza su conjunto de votantes, conformado por las muestras de su soporte. Si una muestra no tuviera muestras en su soporte, se añade al conjunto resultante
5	$mvs \leftarrow \arg\max_{z \in T}\{\lvert Voters_z \rvert\}$	Se busca la muestra que haya recibido mayor cantidad de votos (mvs), y se añade al conjunto resultante
6	$V \leftarrow V \bigcup \{mvs\}$	
7	foreach $z \in Voters_{mvs}$ foreach $q \in T$ $Voters_q \leftarrow Voters_q \setminus \{z\}$	Se actualizara el conjunto de votantes, eliminando a las muestras que se encontraban cercanas al mvs
8	if $\sum_{q \in T} \lvert Voters_q \rvert \neq 0$ then goto 5	Si existen muestras que tienen votos, volver a 5
9	if $\lvert Current \rvert < \lvert V \rvert$ then $Current \leftarrow V$ goto 2	Se inicia una nueva iteración, si se han eliminado algún objeto en la iteración actual
10	end	

4.1.11.2 Valoración

Ventajas: No depende del orden ni del azar. Obtiene un conjunto consistente, que aunque no es siempre el mínimo, sí es de cardinal pequeño.

Desventajas: Por ser un método guiado solamente por la consistencia, el resultado puede no ser representativo del universo en el que están definidos los elementos de la matriz de entrenamiento.

Comentarios: El problema de la obtención de un conjunto consistente mínimo es np completo incluso en presencia de solo dos clases. Esto fue mostrado por Wilfong [56]. Modifica ligeramente la región de decisión.

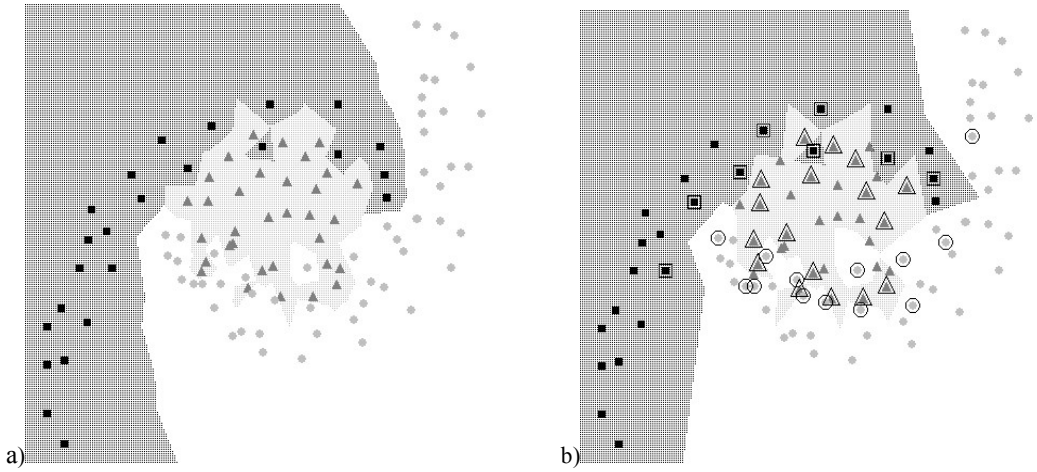

Fig. 28 a) Base de datos "Banana y Pelota". b) MCS aplicado a "Banana y Pelota". Véase la ligera modificación de la región de decisión.

4.1.12 Compact Set Editing (CSE)

Este método fue enunciado por García-Borroto y colaboradores en 2005 [44]. Fue creado para la selección de objetos con datos mezclados e incompletos (MID) y funciones de similaridad no necesariamente métricas. Es importante para entender el algoritmo notar que el conjunto T es utilizado para almacenar los elementos que no han sido procesados, y V los que van a quedar en el resultado. El procedimiento $move(x)$ mueve el objeto x de T a V, mientras que $discard(x)$ elimina a x de T.

4.1.12.1 Algoritmo

Notación y Definiciones:

$G(Vert, C)$ — Subconjunto β_0-compacto del grafo de máxima similaridad (grafo orientado donde cada arco del vértice a al vértice b significa que b es el elemento más β_0-similar de a) descrito por el conjunto de vértices $Vert$ y el conjunto de arcos C

$x \prec y$ — $$x \prec y \Leftrightarrow (e_x < e_y) \vee (e_x = e_y \wedge s_x < s_y) \vee (e_x = e_y \wedge s_x = s_y \wedge |Flags_x| > |Flags_y|)$$

Relación de orden. Si un elemento antecede a otro, es porque es menos importante (según la heurística que utiliza el algoritmo), y debe ser eliminado primero

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow \phi$	
2	foreach $x \in T$ $s'_x \leftarrow \left\lvert \{y \in S(x) : \alpha(x) \neq \alpha(y)\} \right\rvert$ $e_x \leftarrow \left\lvert \{y \in A(x) : \alpha(x) = \alpha(y)\} \right\rvert$ $s_x \leftarrow \left\lvert \{y \in S(x) : \alpha(x) = \alpha(y)\} \right\rvert$ $Flags_x \leftarrow \phi$	Cantidad de sucesores de x de clase diferente Cantidad de antecesores de x de su misma clase Cantidad de sucesores de x de su misma clase $Flags_x$, inicialmente vacío, almacena los objetos que fueron eliminados, y cuya correcta clasificación puede ser garantizada por x
3	$V' \leftarrow \{x \in T : s'_x > 0\}$	
4	if $V' \neq \phi$ then $C \leftarrow C \setminus \{(x,y) \in C : x \in V' \wedge \alpha(x) \neq \alpha(y)\}$ foreach $x \in V'$ execute $move(x)$	Todos los objetos que tienen un sucesor (en el grafo de máxima similaridad) de clase diferente, son adicionados al resultado
5	foreach $x \in T$ if $s_x = 0$ then execute $move(x)$	Si un objeto x tiene $s_x = 0$, tiene que ser adicionado al resultado. De no serlo, el resultado puede ser inconsistente.
6	foreach $x \in T$ $lastFlagged \leftarrow false$ foreach $y \in Flags_x$ if $not\left(y \in \bigcup_{z \in T \setminus \{x\}} Flags_z \right)$ then $lastFlagged \leftarrow true$ if $lastFlagged$ then execute $move(x)$	Si el objeto y marcó a x al ser eliminado, y x es el único que tiene esa marca de los que faltan por ser procesados, entonces x tiene que ser adicionado al resultado, para garantizar la consistencia del resultado
7	$sortAsc(T, \prec)$	
8	execute $discard(T[0])$	El elemento menos importante de los que faltan por evaluar es eliminado. El procedimiento $discard$ asegura la consistencia del resultado
9	if $T \neq \phi$ then goto 5	Se realiza otra iteración, hasta que todos los objetos hayan sido procesados
10	end	

Procedimiento $move(x)$

1	foreach $y \in A(x)$ $s_y \leftarrow \infty$	El elemento *x* va a formar parte del resultado, por lo que la información de quienes lo soportan es irrelevante
2	foreach $y \in S(x)$ $e_y \leftarrow e_y - 1$	Ahora sus sucesores tienen un antecesor menos
3	foreach $y \in T$ $Flags_y \leftarrow Flags_y \setminus Flags_x$	Al quedarse *x*, se asegura la correcta clasificación de los elementos de su $Flags_x$, por lo que se pueden eliminar de todos los demás *Flags*
4	$V \overset{x}{\Leftarrow} T$	Se mueve *x* al resultado, eliminándose los arcos asociados
5	$C \leftarrow C \setminus \{(a,b) \in C : a = x \vee b = x\}$	

Procedimiento $discard(x)$

1	if $s_x \neq \infty$ then foreach $y \in S(x)$ $Flags_y \leftarrow Flags_y \setminus \{x\}$	Para eliminar un objeto, hay que asegurar que el resultado siga siendo consistente. Al eliminarse *x*, si su $s_x \neq \infty$ (su correcta clasificación no ha sido aún asegurada), hay que marcar sus sucesores
2	foreach $y \in S(x)$ $e_y \leftarrow e_y - 1$	Los sucesores tienen un antecesor menos
3	foreach $y \in A(x)$ if $s_y \neq \phi$ then $s_y \leftarrow s_y - 1$	Los antecesores tienen un sucesor menos (excepto los que tienen $s_y \neq \phi$, para los que esta información es irrelevante)
4	$T \leftarrow T \setminus \{x\}$	Se elimina x y sus aristas asociadas
5	$C \leftarrow C \setminus \{(a,b) \in C : a = x \vee b = x\}$	

4.1.12.2 Valoración

Ventajas: Es un método estable, logrando una compactación de alrededor del 50% para todas las bases de datos incluso las que tienen clases muy mezcladas (Fig. 29). Permite trabajar con funciones de similaridad generales, incluso las no simétricas.

Desventajas: No logra niveles de compactación comparables con otros métodos de condensación, incluso en clases totalmente separadas (Fig. 30).

Comentarios: El método utiliza un nuevo concepto para trabajar, que es de consistencia de subclase, a lo que, en opinión de sus autores debe el comportamiento estable.

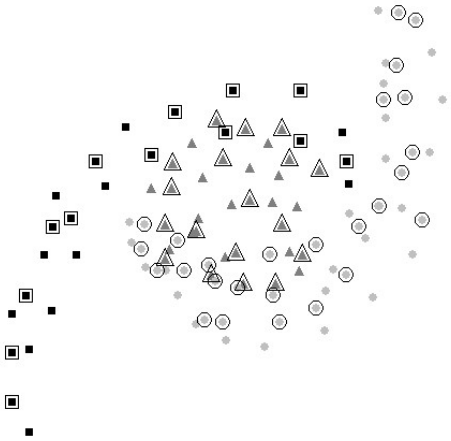

Fig. 29 CSE aplicado a "Banana y Pelota".

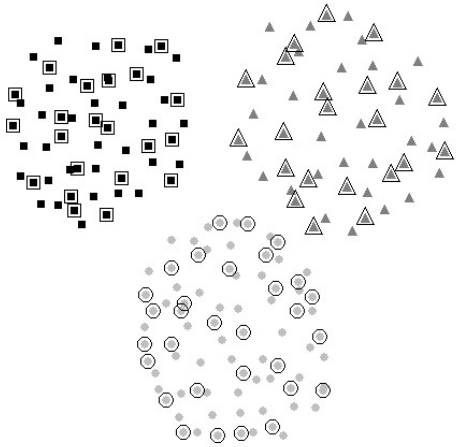

Fig. 30 CSE aplicado a "Bolas".

4.2 Métodos de construcción de objetos

Como se vio en la introducción, existe toda una familia de métodos de construcción de objetos representativos del conjunto original. Estos se basan en varias ideas, como la mezcla de objetos cercanos, el aprendizaje competitivo y los métodos basados en Bootstrap [76]. En general presentan buenos comportamientos, si se escogen de manera correcta sus parámetros (generalmente por prueba y error) y se realizan suficientes iteraciones. Generalmente no se aplican si existen variables cualitativas, por la dificultad de definir las operaciones de suma, multiplicación por un escalar y promedio con sentido en estos dominios.

Chang en 1974 [77] fue el primer creador de un método de construcción de objetos. En este método se substituyen de manera iterativa los dos objetos más cercanos de la misma clase por su media ponderada, si al hacerlo no se degrada la eficacia del clasificador. La ponderación ocurre con los pesos asociados a los objetos, que son inicializados en 1 y se suman para obtener el peso resultante al ocurrir una mezcla. Esto garantiza que los objetos resultantes de varias mezclas, se "muevan" menos de su ubicación actual. Este proceso, que lleva asociado un alto costo computacional, es optimizado por el propio autor de varias maneras. Debido a que la mezcla de objetos es guiada sólo por la consistencia, el resultado final puede no ser representativo del conjunto original [78].

En 1998, Bezdek *et al*. [79], propusieron un nuevo método basado en el método de Chang, denominado MCA (Modified Chang Algorithm). Una de las modificaciones introducidas es la utilización de un promedio simple entre objetos y no una media ponderada. Además, se alteró la búsqueda de candidatos para la mezcla mediante el particionado de la matriz de distancias original en submatrices de clase común. De esta forma se disminuye considerablemente la cantidad de comparaciones que se realizan para buscar los objetos que son más cercanos.

Mollineda *et al*. en el 2002 [78] introdujeron a su vez una variante de MCA, substituyendo la mezcla de objetos por la mezcla de agrupamientos (clusters). Los objetos resultantes del método son los centroides de los grupos resultantes. En el método se utiliza la desigualdad triangular, para acelerar la búsqueda, por lo que sólo es aplicable a espacios métricos.

El aprendizaje competitivo tiene varios representantes como Learning Vector Quantization (LVQ) [80], Decision Surface Mapping (DSM) [81], Learning Vector Quantization with Training Counters (LVQ-TC) [82] y Vector Quantization (VQ) [83]. De forma general se basan en la idea de la selección aleatoria de un subconjunto inicial, y su modificación en varias iteraciones. Generalmente utilizan la estrategia del "ganador lo toma todo", siendo el objeto más cercano a cada objeto del conjunto inicial desplazado. La dirección del desplazamiento de los objetos coincide con la línea recta que une al prototipo con el objeto analizado. La dirección se determina según las clases de ambos objetos, si son iguales se acercan, si son diferentes se alejan.

Hamamoto *et al*. en 1997 [84] desarrollaron cuatro métodos de Bootstrap. En estos métodos los objetos son construidos por la media de los *k* vecinos más cercanos de su misma clase de n objetos seleccionados al azar (sin repeticiones). El usuario define el número de intentos que se realizarán, de los cuales se selecciona el resultado con un menor error. El Bootstrap es utilizado en la estadística moderna para la estimación de las distribuciones de muestreo de un estimador, mediante el re-muestreo con reemplazo de la muestra original

4.2.1 Clustering and relabeling

Este método permite utilizar cualquier algoritmo de clasificación no-supervisada, que genere centroides de los agrupamientos, para la búsqueda de objetos. Este método puede ser útil si los centroides son 'buenos' representantes de sus agrupamientos. Finalmente se utiliza un procedimiento de etiquetado de objetos.

Un método de clasificación no supervisada bueno para este propósito es el C-Means, que fue el que utilizamos en nuestra implementación.

4.2.1.1 Algoritmo

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow clustering(T)$	Inicialmente el resultado es el centro de los clusters
2	execute $label(V,T)$	Se le asignan finalmente las clases a los objetos de V , con el procedimiento $label$
3	end	

Procedimiento $label(V,T)$

1	foreach $v \in V$ $Winnners_v \leftarrow \phi$	El conjunto $Winnners_v$ guarda los objetos de T para los que v fue su vecino más cercano
2	foreach x *in* T	Se adiciona x al $Winners$

	$v \leftarrow neighbor_1(x,V)$ $Winners_v \leftarrow Winners_v \bigcup \{x\}$	de su objeto más cercano en V
3	foreach v *in* V $\alpha(v) \leftarrow \arg\max_{K_i} \{\lvert y \in Winners_v : \alpha(y) = K_i \rvert\}$	A cada objeto se le asigna la clase mayoritaria entre sus $Winners$, lo que garantiza cometer la menor cantidad de errores al clasificar

4.2.1.2 Valoración

Ventajas: permite la aplicación de cualquier método de agrupamiento para la construcción de objetos, pudiéndose aprovechar el enorme arsenal de herramientas existentes en la materia.

Comentarios: Dependen fundamentalmente del método de agrupamiento escogido. En el caso del Hard C-Means se pueden obtener resultados muy diferentes en dos ejecuciones con los mismos datos. Su extensión para datos mezclados e incompletos depende del algoritmo de agrupamiento a utilizar.

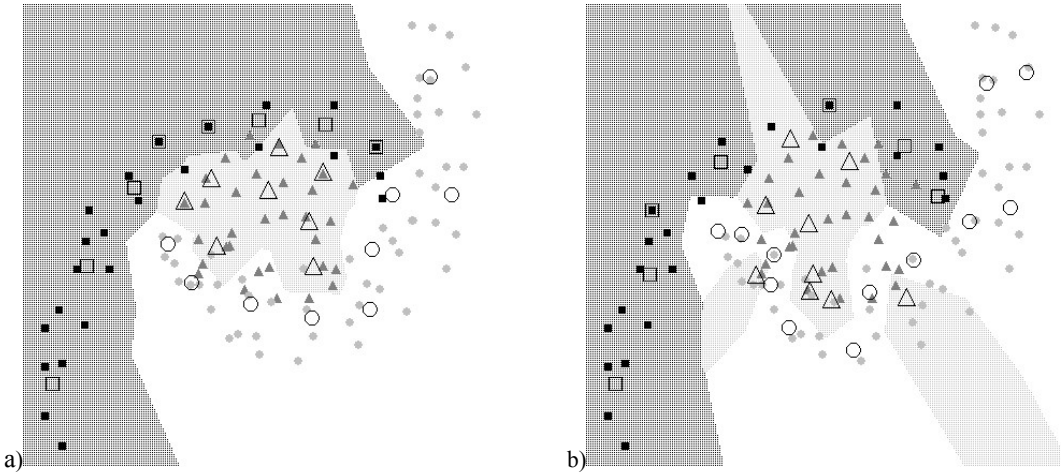

Fig. 31 Dos ejecuciones del Clustering and Relabeling con C-Means aplicado a "Banana y Pelota". (Número de clusters=30).

4.2.2 Bootstrap 4 (BT4)

Hamamoto y colaboradores en 1997 [84] desarrollaron cuatro métodos de Bootstrap, de los cuales analizaremos en este trabajo el Bootstrap 4. Los objetos son construidos por la media de los *k* vecinos más cercanos de su misma clase, según se definen a continuación, de $\mathbf{n_p}$ objetos seleccionados al azar (sin repeticiones). El usuario define el número de intentos que se realizarán, de los que selecciona el mejor.

4.2.2.1 Algoritmo

Notación y Definiciones:

$(U,+,\cdot)$ Espacio vectorial definido sobre U, con la suma entre dos objetos y el producto por un escalar

Entradas:

n_{K_i} Número de objetos a construir en la clase K_i

T Matriz de entrenamiento

k Número de vecinos a considerar en el promedio

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow \phi$	
2	foreach $i \in \{1,\ldots,l\}$ $NewV \leftarrow randomSC(T, K_i, n_{K_i})$ foreach $x \in NewV$ $NN \leftarrow neighborsSC_k(x,T)$ $bt \leftarrow mean(NN)$ $V \leftarrow V \bigcup \{bt\}$	Se seleccionan n_{k_i} muestras aleatorias. Se adiciona al resultado el centroide (calculado con la media aritmética) de los k vecinos más cercanos a x
3	end	

4.2.2.2 Valoración

Ventajas: Es muy simple y rápido. Según Kuncheva [46] la familia de Bootstrap permite obtener muy buenos resultados. En experimentos realizados por los autores tuvo un comportamiento excelente.

Desventajas: Como todos los esquemas aleatorios depende fuertemente del azar (Fig. 32). Presupone la existencia de un espacio donde se defina una suma entre objetos y el producto de un objeto por un escalar para que las muestras puedan ser promediadas.

Comentarios: De todos los esquemas aleatorios que fueron probados, este es sin dudas el superior, incluso en bases de datos grandes. Podría ser extendido para trabajar con datos mezclados e incompletos, substituyendo la media aritmética de los objetos por un concepto afín, como por ejemplo, el holotipo y modificando la medida de similaridad y el clasificador.

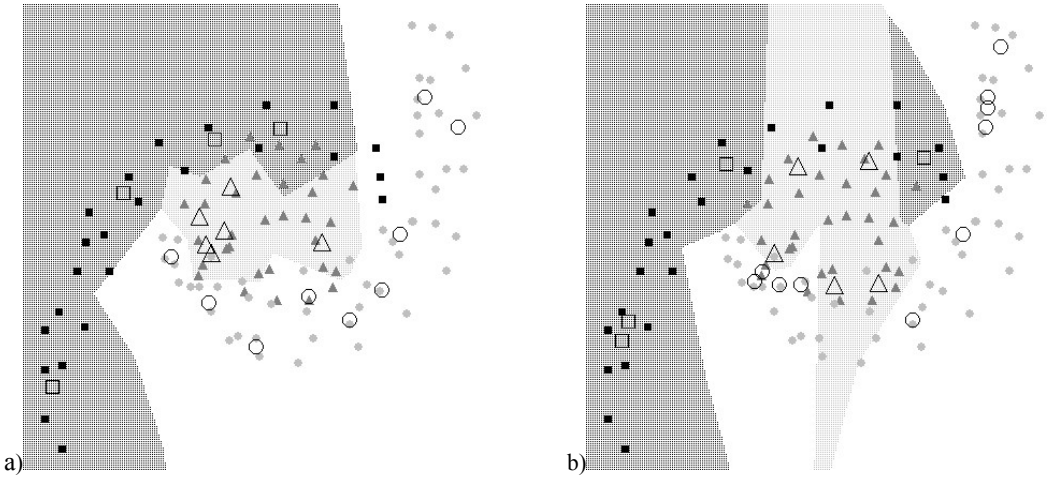

Fig. 32. Dos aplicaciones del BTS-3 (np=20, ni=10) a Bananas y Pelota.

4.2.3 Prototypes for Nearest Neighbor (PNN)

Este algoritmo fue propuesto por Chang en 1974 [77]. Trata de mezclar los objetos más cercanos de la misma clase siempre que esta mezcla no degrade el desempeño del clasificador, considerando el 1-NN. La descripción algorítmica que se brinda se ha descrito de forma diferente a la del artículo original, con el propósito de facilitar la comprensión del funcionamiento de este método.

4.2.3.1 Algoritmo

Notación y Definiciones:

$(U,+,\cdot)$ Espacio vectorial definido sobre U, con la suma entre dos objetos y el producto por un escalar

Entradas:

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$B \leftarrow T$	
2	$V \leftarrow \phi$	
3	foreach $y \in B$ $Weight_y \leftarrow 1$	A cada elemento de B se le asigna peso 1
4	$V \overset{First(B)}{\Leftarrow} B$	Se mueve el primer elemento de B hacia V
5	$merge \leftarrow false$	La bandera $merge$ controla si ocurrió alguna mezcla de objetos en la iteración actual

6	$(x,y) \leftarrow \arg\min_{\substack{a\in V \\ b\in B}} \{d(a,b)\}$	Se seleccionan los objetos más cercanos entre sí, uno en V y el otro en B
7	if $\alpha(x) \neq \alpha(y)$ then $V \overset{y}{\Leftarrow} B$ else $p^* \leftarrow \frac{weight_x \cdot x + weight_y \cdot y}{weight_x + weight_y}$ $lastError \leftarrow \varepsilon_{V\cup B}^{kNN}(T)$ $currentError \leftarrow \varepsilon_{(V\setminus\{x\})\cup(B\setminus\{y\})\cup\{p^*\}}^{kNN}(T)$ if $lastError < currentError$ then $V \overset{y}{\Leftarrow} B$ else $V \leftarrow V \setminus \{x\}$ $B \leftarrow B \setminus \{y\}$ $V \leftarrow V \cup \{p*\}$ $weight_{p*} \leftarrow weight_x + weight_y$ $merge \leftarrow true$	Si son de clases diferentes se mueve y para V Si son de la misma clase se construye un nuevo objeto p^*, mediante el promedio ponderado de x y y Error actual Error si se substituyen x y y por p^* Si aumenta el error con el nuevo objeto, no se hace acepta la mezcla. Si el error no aumenta, x y y son substituidos por p^*, y el peso del nuevo objeto es calculado
8	if $B \neq \phi$ then goto 6	Iterar hasta que B sea vaciado
9	if $merge$ then $B \leftarrow V$ $V \leftarrow \phi$ goto 4	Si se ha realizado alguna mezcla, se repite el proceso
10	end	

4.2.3.2 Valoración

Ventajas: Puede condensar muy sensiblemente la matriz original.

Desventajas: Se inicializa tomando el primer elemento de la matriz de entrenamiento, modificándose sensiblemente el resultado entre dos corridas del algoritmo con idénticos datos pero en diferente orden (Fig. 33). Según [85] este método presenta dos desventajas fundamentales:

- La estrategia de mezcla de un par de objetos basada sólo en la consistencia es muy restringida, por lo que puede dar resultados alejados del óptimo, tanto desde el punto de vista de tamaño, como de representatividad.

- Emplea una cantidad considerable de tiempo de cómputo para el chequeo exhaustivo de las posibles mezclas.

Comentarios: Obtiene un conjunto consistente. En el artículo original [77] se detalla un conjunto importante de optimizaciones para disminuir el tiempo de ejecución.

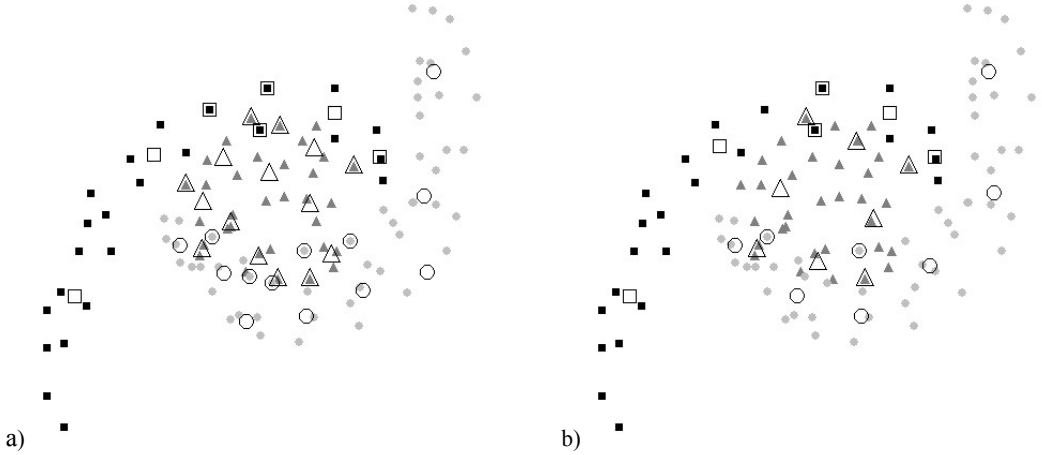

Fig. 33. Dos resultados de la búsqueda con PNN en "Banana y Pelota".

4.2.4 Learning Vector Quantization (LVQ – 1)

La familia de los LVQ incluye un largo espectro de esquemas de *aprendizaje competitivo* [80]. Las variantes *presupervisadas* ajustan los objetos según el esquema de premio – castigo, de manera que el objeto más cercano a cada objeto de la matriz de entrenamiento, si es de la misma clase es acercado, y si es de clase contraria, alejado.

Inicialmente, se seleccionan np elementos de la matriz de entrenamiento de forma aleatoria, (siendo *np* un parámetro definido por el usuario) para obtener un conjunto de objetos **V.** Cada clase está representada por al menos un objeto.

Además, el usuario define tres parámetros: la tasa de aprendizaje $l_r \in [0,1]$ que indica cuánto se va a mover el objeto, una constante de variación de la tasa de aprendizaje $\eta \in [0,1]$ y el número de iteraciones n_i. La tasa de aprendizaje indica cuánto se va a "mover" el objeto y la variación de la tasa de aprendizaje, como su nombre lo indica, determina la disminución progresiva que va a sufrir la tasa de aprendizaje, haciendo que cada vez los objetos se "muevan" menos. Se utiliza un aprendizaje competitivo para actualizar la matriz de entrenamiento. Sea el vector v el objeto más cercano de la entrada x la estrategia de actualización es:

$v = v + l_r(x - v)$ si $\alpha(v) = \alpha(x)$

$v = v - l_r(x - v)$ si $\alpha(v) \neq \alpha(x)$

Además se actualiza la tasa de aprendizaje $li = \eta \cdot li$ para cada pasada del algoritmo y se permuta la matriz de entrenamiento. El algoritmo finaliza cuando no hay malas clasificaciones (utilizando un clasificador 1-NN) o cuando se llega al número máximo de iteraciones *ni*.

4.2.4.1 Algoritmo

Notación y Definiciones:

$(U,+,\cdot)$ Espacio vectorial definido sobre U, con la suma entre dos objetos y el producto por un escalar

Entradas:

np Número de objetos

$lr \in [0,1]$ Tasa de aprendizaje

$\eta \in [0,1]$ Variación de la tasa de aprendizaje

ni Número de iteraciones

T Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow randomSCP(T,np)$	
2	$it \leftarrow 1$	
3	$l \leftarrow lr$	
4	foreach $x \in T$ $v \leftarrow neighbor_1(x,V)$ if $\alpha(v) = \alpha(x)$ then $v \leftarrow v + l(x - v)$ else $v \leftarrow v - l(x - v)$ $l \leftarrow \eta \cdot l$	Para cada objeto, se halla su objeto más cercano Si el objeto de V más cercano a x es de su misma clase, entonces es acercado; en caso contrario, es alejado. Este desplazamiento ocurre sobre la línea que une ambos objetos, y tiene magnitud l Se disminuye la tasa de aprendizaje
5	if $(\varepsilon_V^{kNN}(T) \neq 0) \wedge (it < n_i)$ then $permute(T)$ $it \leftarrow it + 1$ goto 3	Si existen elementos mal clasificados con el 1-NN y no se ha alcanzado aún el número de iteraciones, se permutan los elementos de T, y se inicia otra iteración. Esta permutación disminuye la dependencia del método del orden de los objetos.
6	end	

4.2.4.2 Valoración

Ventajas: En experimentos realizados por los autores, este método ha obtenido en muchos casos bajos errores con una matriz de control. Es uno de los LVQ más robustos [86], y la tasa de aprendizaje puede ser optimizada aproximadamente para una rápida convergencia.

Desventajas: La desventaja fundamental de los LVQ [87] radica en que son muy fuertemente dependientes del proceso de inicialización (Por ejemplo, la selección inicial aleatoria de los objetos y el orden de los mismos, Fig. 34). Por otra parte usualmente terminan cuando se estabilizan los conjuntos de objetos resultantes, en vez de utilizar un criterio relacionado con la calidad de *V* como representación de *T*, por lo que puede alterarse significativamente la región de decisión del clasificador.

Comentarios: Una amplia descripción de los LVQ puede ser encontrada en [86], donde se comparan 3 variantes. El LVQ1 es recomendado en la mayoría de los casos por su rápida convergencia. En muchas ocasiones el LVQ1 da resultados satisfactorios, y no es necesario aplicar las otras variantes, más lentas y complejas.

Los LVQ, al igual que muchas redes neuronales, sufren de problemas de generalización si son sobre entrenados. Es por esto que se hace necesario detener el proceso de entrenamiento después de un número 'óptimo' de pasos, por ejemplo, de 50 a 200 veces el número total de vectores (en dependencia del algoritmo y las tasas de aprendizaje). Este valor puede ser encontrado sólo por la experiencia, y también depende de los datos.

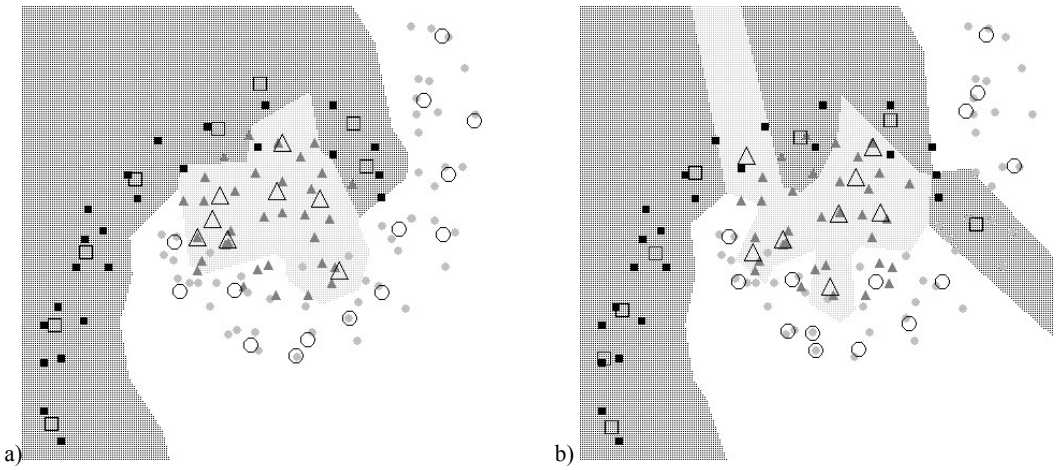

Fig. 34. Dos ejecuciones del LVQ1 aplicado a "Banana y Pelota". (np=30, ni=100, lr=0.8, η=0.8). Nótese la dependencia del orden y del azar.

4.2.5 Decision Surface Mapping (DSM)

Es una variante de los LVQ desarrollada por Geva y Sitte en 1991 [81], que se diferencia de ellos en que el premio – castigo sólo se aplica en el caso de una mala clasificación del elemento actual con respecto a los objetos en el conjunto resultante. También está concebido para utilizar un clasificador 1-NN.

4.2.5.1 Algoritmo

Notación y Definiciones:

$(U,+,\cdot)$	Espacio vectorial definido sobre U, con la suma entre dos objetos y el producto por un escalar

Entradas:

np y ni	Número de objetos y de iteraciones, respectivamente
$lr \in [0,1]$	Tasa de aprendizaje
$\eta \in [0,1]$	Variación de la tasa de aprendizaje
T	Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow randomSCP(T, np)$	
2	$it \leftarrow 1$	
3	$l \leftarrow lr$	
4	foreach $x \in T$ $v \leftarrow neighbor_1(x, V)$ if $\alpha(v) \neq \alpha(x)$ then $v \leftarrow v - l \cdot (x - v)$ $y \leftarrow neighborSC_1(x, V)$ $y \leftarrow y + l \cdot (x - y)$ $l_i \leftarrow \eta \cdot l_i$	Si el objeto v más cercano a x no es de su misma clase, se aleja a v de x. Después se acerca a x el objeto en V más cercano. La taza de aprendizaje se actualiza
5	if $(\varepsilon_V^{k-NN}(T) \neq 0) \wedge (it < n_i)$ then $permute(T)$ $it \leftarrow it + 1$ goto 3	Si existen elementos mal clasificados con el 1-NN y no se ha alcanzado aún el número de iteraciones, se permutan los elementos de T, y se inicia otra iteración. Esta permutación disminuye la dependencia del método del orden de los objetos.
6	end	

4.2.5.2 Valoración

Ventajas: Según sus autores el DSM brinda mejores aproximaciones a los bordes de clasificación. Esto sucede debido a que los objetos de la frontera (generalmente mal clasificados), "halan" a los objetos de su clase del conjunto resultante, por lo que éstos se aproximan a las fronteras de decisión.

Desventajas: Según sus autores el DSM es menos estable que los LVQ. Al igual que los LVQ depende del azar y del orden de los objetos en la matriz de entrenamiento. También puede alterar significativamente la frontera de decisión del clasificador (Fig. 35).

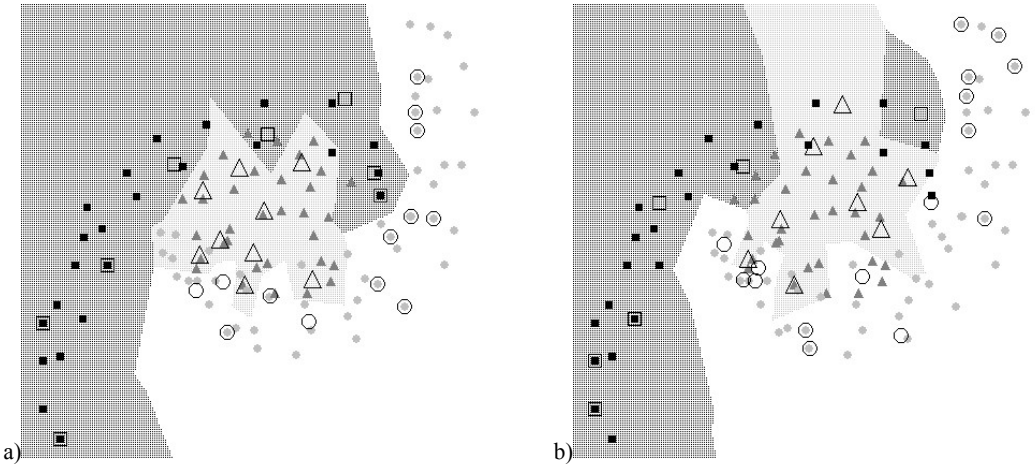

Fig. 35. Dos ejecuciones del DSM aplicado a "Banana y Pelota". (np=30, ni=100, lr=0.8, η=0.8).

4.2.6 Learning Vector Quantization with Training Counters (LVQTC)

Este método, creado por Odorico [82] en 1997, es similar al LVQ1, pero el ajuste de los objetos va a depender de dos factores:

1. La distancia existente del objeto a un determinado elemento de la matriz de entrenamiento.
2. El historial del objeto.

La importancia histórica de un objeto es determinada por el número de veces que ha sido el ganador. El procedimiento es inicializado de la misma forma que el LVQ1, con la adición de un contador de entrenamiento (TC) vació con *c* capacidades, una para cada clase. Cada vez que para un objeto *x* se selecciona un objeto ganador *t*, el contador de entrenamiento de *t* se incrementa para la clase a la que pertenece *x*.

La ecuación de premiado de los objetos utiliza el TC para garantizar que los objetos que más veces han sido ganadores, sean menos desplazados hacia una nueva posición, porque deben haber ya alcanzado una posición adecuada. Según Bezdek [46] esta estrategia puede ser catalogada como 'premio decreciente'.

4.2.6.1 Algoritmo

Notación y Definiciones:

$(U,+,\cdot)$	Espacio vectorial definido sobre U, con la suma entre dos objetos y el producto por un escalar

Entradas:

np y ni	Número de objetos y de iteraciones, respectivamente
$lr \in [0,1]$	Tasa de aprendizaje
$\eta \in [0,1]$	Variación de la tasa de aprendizaje
p	Umbral
T	Matriz de entrenamiento

Salida:

V Resultado del método

Pasos:

1	$V \leftarrow randomSCP(T, np)$	
2	$it \leftarrow 1$	
3	foreach $v \in V$ foreach $i \in \{1,\ldots,l\}$ $hist_{v,K_i} \leftarrow 0$ $Calls_v \leftarrow \phi$	El histórico por objeto por clase se inicializa en 0 y las llamadas en vacío.
4	foreach $x \in T$ $v \leftarrow neighbor_1(x,V)$ $hist_{v,\alpha(x)} \leftarrow hist_{v,\alpha(x)} + 1$ $Calls_v \leftarrow Calls_v \cup \{x\}$ if $\alpha(v) = \alpha\{x\}$ then $tc \leftarrow \lvert Calls_v \rvert$ $v \leftarrow v + \frac{l_r}{tc}(x - v)$ $l_r \leftarrow \eta \cdot l_r$	Para cada objeto x, se busca el objeto v más cercano, y se actualiza el histórico y $Calls$. Si coincide la clase del elemento con la de su objeto más cercano, éste se actualiza, proporcional al valor de la taza de aprendizaje, y de las veces que haya sido movido anteriormente ($\lvert Calls_v \rvert$) Se actualiza la tasa de aprendizaje
5	foreach $v \in V$ if $\lvert Calls_v \rvert < p$ then $V \leftarrow V \setminus \{x\}$	Si un objeto no ha sido el más cercano un número mayor de veces que el umbral establecido, es eliminado
6	$Vn \leftarrow \phi$	
7	foreach $v \in V$ $mcc \leftarrow \max_{i \in \{1,\ldots,l\}} \{hist_{v,K_i}\}$ $C \leftarrow \arg\max_{i \in \{1,\ldots,l\}} \{hist_{v,K_i}\}$ if $(\alpha(v) \neq C) \wedge (mcc > p)$ then $Vsc \leftarrow \{v_{sc} \in Calls_v : \alpha(v_{sc}) = C\}$ $v_{sc} \leftarrow mean(Vsc)$ $\alpha(v_{sc}) \leftarrow C$ $Vn \leftarrow Vn \cup \{x\}$ else $Vn \leftarrow Vn \cup \{v\}$	Si la clase de un objeto no coincide con la clase mayoritaria de su $Calls$, este es substituido por el promedio de los elementos de dicha clase, asignándole ésta como su clase.
8	$V \leftarrow Vn$	

9	if $(it < n_i) \wedge (\varepsilon_V^{kNN}(T) \neq 0)$ then $permute(T)$ $it \leftarrow it + 1$ goto 3	Si existen elementos mal clasificados con el 1-NN y no se ha alcanzado aún el número de iteraciones, se permutan los elementos de T, y se inicia otra iteración. Esta permutación disminuye la dependencia del método del orden de los objetos.
10	end	

4.2.6.2 Valoración

Desventajas: Depende del azar y necesita un espacio donde se defina una suma entre objetos y un producto de un objeto por un escalar. Puede modificar sensiblemente las regiones de decisión del clasificador (Fig. 36).

Comentarios: Aunque este algoritmo tiene un fuerte componente heurístico y depende críticamente del la selección del umbral, tiene la ventaja que los objetos resultantes pueden ser etiquetados de manera no dura (o difusa), utilizando los TC.

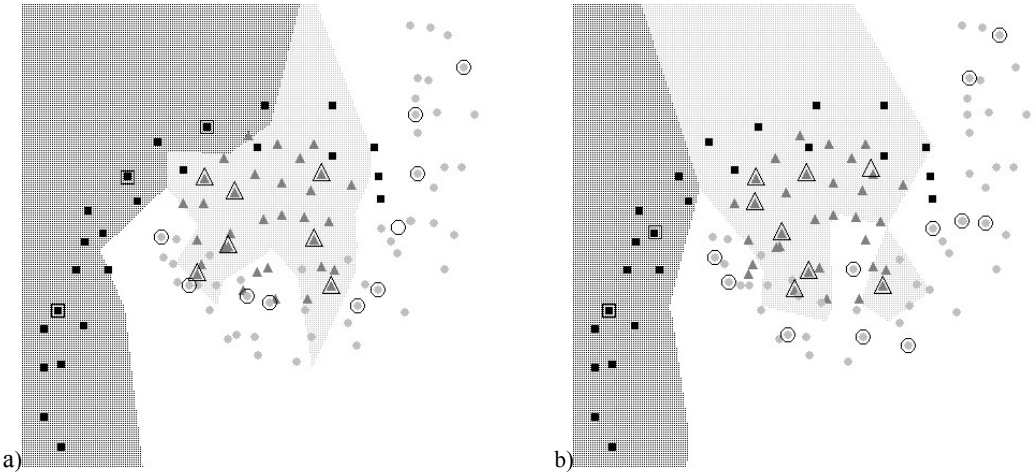

Fig. 36. Dos ejecuciones del LVQTC aplicado a "Banana y Pelota". (np=30, T=10, α=0.8, η=0.3, p=3).

4.2.7 Vector Quantization (VQ)

Es un método *postsupervisado* de construcción de objetos. El usuario define el número de objetos que desea obtener (*np*), el número máximo de iteraciones (*ni*), la tasa de aprendizaje (*lr*) y la medida de cercanía entre dos conjuntos de objetos generados en iteraciones sucesivas (*err*). Los objetos, inicialmente sin clase, se van premiando cuando sean los más cercanos y la tasa de aprendizaje se actualiza para cada iteración. El proceso termina cuando se alcanza el número máximo de iteraciones o se satisface que la medida de cercanía entre iteraciones sucesivas es menor que un cierto umbral, definido por el usuario. Finalmente se realiza un proceso de etiquetado de los objetos resultantes, calculando para cada objeto las muestras que gana por

clase, y asignándole la clase mayoritaria. De esta manera se garantiza un mínimo de malas clasificaciones del conjunto resultante [46].

4.2.7.1 Algoritmo

Notación y Definiciones:

$(U,+,\cdot)$	Espacio vectorial definido sobre U, con la suma entre dos objetos y el producto por un escalar

Entradas:

np	Número de objetos
$\eta \in [0,1]$	Variación de la tasa de aprendizaje
$lr \in [0,1]$	Tasa de aprendizaje
ni	Número de iteraciones
err	Variación entre los objetos de dos iteraciones sucesivas que detiene la ejecución del algoritmo
T	Matriz de entrenamiento

Salida:

V	Resultado del método

Pasos:

1	$V \leftarrow randomS(T,np)$	Conjunto inicial de objetos seleccionados aleatoriamente
2	$it \leftarrow 1$	
3	$l \leftarrow lr$	
4	$Vant \leftarrow V$	
5	foreach $x \in T$ $v \leftarrow neighbor_1(x,V)$ $v \leftarrow v + l(x - v)$ $l \leftarrow \left(1 - \frac{it}{ni}\right) lr$	Para cada objeto de la matriz de aprendizaje, se busca su objeto más cercano en V, y éste es acercado a aquel y se actualiza su valor con la ecuación correspondiente
6	$s \leftarrow \sum_{V} d(v,v')$ if $(it < ni) \wedge (s > err)$ then $it \leftarrow it + 1$ goto 3	El objeto $v' \in V$ es el generado a partir de movimientos de $v \in Vant$ Si la distancia es superior al valor de corte establecido y no se ha alcanzado el número de iteraciones, se regresa a 3
7	execute $label(V,T)$	Se asignan las clases a los objetos finales
8	end	

Procedimiento $label(V,T)$

1	foreach $v \in V$ $Winnners_v \leftarrow \phi$	El conjunto $Winnners_v$ guarda los objetos de T para los que v fue su vecino más cercano
2	foreach x *in* T $v \leftarrow neighbor_1(x,V)$ $Winners_v \leftarrow Winners_v \bigcup \{x\}$	Se adiciona x al $Winners$ de su objeto más cercano en V
3	foreach v *in* V $\alpha(v) \leftarrow \arg\max_{K_i} \{\lvert y \in Winners_v : \alpha(y) = K_i \rvert\}$	A cada objeto se le asigna la clase mayoritaria entre sus $Winners$, lo que garantiza cometer la menor cantidad de errores al clasificar

4.2.7.2 Valoración

Desventajas: Depende del azar y al igual que otros algoritmos de aprendizaje competitivo puede provocar cambios importantes en la frontera de decisión del clasificador (Fig. 37).

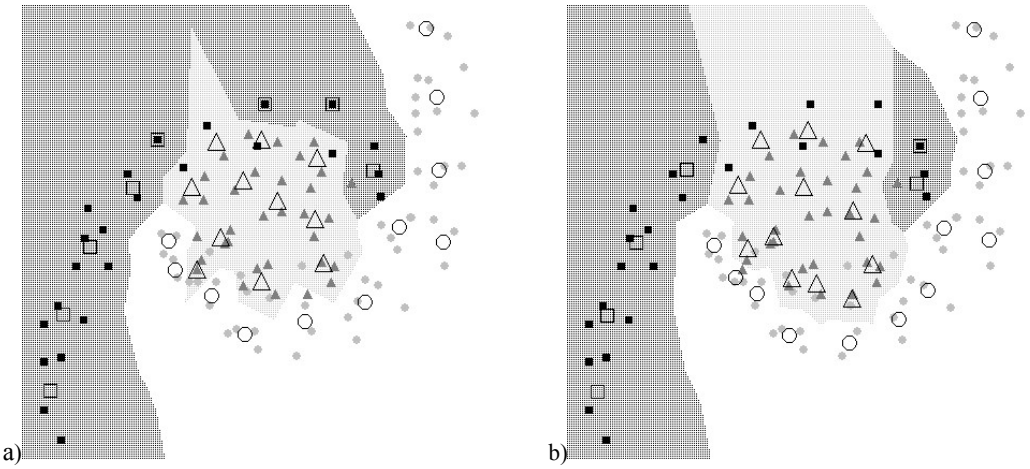

Fig. 37. Dos ejecuciones del VQ aplicado a "Banana y Pelota". (np=30, *ni*=100, *lr*=0.8, *err*=0.8). Nótese la dependencia del orden y del azar.

5 Taxonomía de los métodos de selección y construcción de objetos

En esta sección presentaremos cuatro taxonomías de los métodos de búsqueda de objetos, creadas y enriquecidas por diferentes autores. Algunas de ellas son subjetivas, y otras no cubren todas las posibilidades, pero consideramos que son una herramienta eficaz para el entendimiento y la posterior selección del método más adecuado para un problema determinado.

5.1 Selección del número de objetos (T1)

5.1.1 Por el usuario (np-usuario)

El número de objetos se define *a priori* por el usuario. Esta variante tiene la ventaja obvia que fuerza la determinación de un número pequeño de objetos. Sin embargo, esto limita la complejidad del problema que se aborda [64]. Por otra parte, si se considera que cada objeto es un grado de libertad más del sistema, esta variante puede favorecer modelos más simples de solución del problema.

5.1.2 Por el sistema (np-auto)

Este es un esquema preferido muchas veces por el usuario, ya que realizar la selección *a-priori* del número de objetos no es una tarea trivial. En general estos métodos no encuentran un conjunto objetos mejores, pero el usuario tiene un parámetro menos que escoger. Usualmente substituyen el proceso de prueba y error que realiza el usuario para encontrar el "mejor" conjunto de objetos, basándose en algunos criterios matemáticos o heurísticos para seleccionar np. [46]

5.1.3 Adaptativo (np-adaptativo)

El usuario especifica inicialmente el número tentativo de objetos, pero el método puede aumentar o disminuir posteriormente este número.

5.2 Presupervisado vs postsupervisado (T2)

5.2.1 Presupervisado

Se utilizan las etiquetas de los datos *durante* el entrenamiento para guiar un algoritmo hacia "buenos" objetos (ya etiquetados). La intuición sugiere que estos métodos son superiores, porque utilizan la información adicional contenida en las clases para realizar la búsqueda. Esto ha sido parcialmente mostrado en algunos trabajos. [46]

5.2.2 Postsupervisado

Las etiquetas son ignoradas durante el entrenamiento, siendo usadas *a posteriori* para etiquetar los objetos. Permite utilizar todas las herramientas creadas en el análisis de clusters en la búsqueda.

5.3 Meta (T3)

Como se había comentado antes, existen dos motivaciones fundamentales para buscar objetos de una matriz de entrenamiento:

1. Reducir el tamaño del la matriz de entrenamiento, en búsqueda de la eficiencia. Son conocidos como métodos de condensación y poseen como un subconjunto importante de ellos los que buscan calcular el mínimo conjunto consistente.
2. Editar para mejorar la eficacia del clasificador (o edición basada en el error). Estos trabajos, comenzados por Wilson [11], poseen una base estadística fuerte. En ellos se enuncian demostraciones para distribuciones particulares de los objetos, o para el caso asimptótico.

Otros trabajos no pueden ser clasificados estrictamente en una de estas dos categorías, pues tratan de hallar un compromiso racional entre ambas metas.

5.4 Paradigma (T4)

Esta es, quizás, la más subjetiva de todas las taxonomías, pero, al mismo tiempo, una de las más útiles para entender cómo funcionan los métodos. Aunque no existe una terminología única universalmente aceptada, en este trabajo se utilizan los términos definidos en el libro "Combining Pattern Classifiers" de Ludmila Kuncheva [47].

5.4.1 Selección de objetos

En los métodos de selección, los objetos resultantes son seleccionados de la matriz de entrenamiento. Ejemplos de este tipo de métodos lo constituyen el CNN, el ENN, entre otros.

5.4.1.1 Selección guiada por la consistencia

La consistencia respecto al conjunto de entrenamiento ha sido un criterio ampliamente utilizado para la búsqueda de objetos, aunque posee varios inconvenientes considerables:

- Asegura la buena clasificación de todos los elementos de la matriz de entrenamiento, pero no garantiza, en general, que no ocurran cambios muy grandes en las regiones de decisión, lo que es más importante para garantizar el poder de generalización del clasificador.
- Wilfong [56] mostró que el problema de encontrar un subconjunto consistente de tamaño mínimo es np-completo en presencia de 3 o más clases. Posteriormente él mismo mostró que incluso para dos clases, el proceso selectivo es np-completo. Debido a su diseño, estos métodos no pueden negociar un poco de eficacia para ganar en eficiencia. Tal mecanismo es muy deseable ya que a costa de una o dos malas clasificaciones se puede reducir a la mitad el número de objetos [47].

En esta categoría se incluyen algunos métodos que, aunque no obtienen subconjuntos consistentes, se guían por la no disminución de la eficacia del clasificador. Por lo general incluyen al inicio un método de edición basada en el error, para eliminar los objetos ruidosos, y por eso el resultado no es consistente.

5.4.1.2 Edición basada en el error

Los métodos de edición basada en el error, o como comúnmente se les llama, simplemente métodos de edición, asumen que los objetos cuyos vecinos son de clase contraria son "ruidosos", y por tanto deben ser eliminados. No están algorítmicamente conectados explícitamente con los errores de resubstitución ni de generalización. Estos métodos no tienen un mecanismo explícito para limitar el número de objetos o para penalizar los conjuntos de alta cardinalidad. Usualmente se obtienen muchos más objetos que los necesitados. [47]

5.4.1.3 Almacenado de objetos típicos

Una estrategia creada por Zhang [66], trata de calcular un índice de tipicidad de cada objeto dentro de su clase. Basado en este índice puede tomarse una decisión de qué objetos permanecerán, y cuáles serán removidos. Nótese que esta estrategia puede ser utilizada tanto para condensar (quedándose con los más típicos), como para eliminar los objetos "ruidosos" (sin dudas, los menos típicos). Otro esquema, propuesto por Paredes y Vidal [88], realiza la decisión basado en el peso de cada objeto, que es calculado de manera similar.

5.4.1.4 Explotar el conocimiento del dominio

Estos métodos utilizan algunas heurísticas basadas en el conocimiento que se tiene *a priori* del dominio del problema, para decidir qué objetos serán seleccionados, y cuáles serán descartados. Muchas veces la selección inicial de la matriz de entrenamiento es realizada por un experto, que aplica intuitivamente un esquema similar.

5.4.1.5 Optimización

Estos métodos usan técnicas de optimización aproximada para encontrar un subconjunto lo más cercano posible al mínimo consistente.

5.4.1.6 Generación de la frontera de decisión

Consideremos dos elementos *x, y* con $\alpha(x) \neq \alpha(y)$. Si los dos puntos son usados en una regla 1-NN ellos determinan una frontera de decisión que es el hiperplano que ortogonalmente biseca el segmento de recta que une ambos puntos. De esta forma cuando un conjunto de la matriz de entrenamiento es seleccionado, dichos hiperplanos se seleccionan implícitamente. Sin embargo, es posible seleccionar estos hiperplanos explícitamente de la misma forma. Cuando los clasificadores están diseñados para manipular más hiperplanos que clases de patrones, estos son llamados clasificadores lineales por trozos (piece-wise) [89].

También es posible generar fronteras de decisión no-paramétricas con otras superficies además de los hiperplanos. Priebe [90, 91] modela la frontera de decisión utilizando bolas.

5.4.1.7 Grafos de proximidad

Los grafos de proximidad (diagramas de Voronoi, grafo de vecinos relativos, grafo de Gabriel) han sido ampliamente utilizados para la selección de objetos [43]. Estos métodos generan inicialmente el grafo de proximidad pertinente (lo que puede ser un proceso costoso computacionalmente en tiempo y uso de memoria), y lo utilizan para seleccionar objetos.

5.4.1.8 Esquemas aleatorios y evolutivos

Estos esquemas tienen en común un fuerte componente aleatorio, con muchos parámetros a seleccionar por el usuario. El más sencillo de todos es la selección aleatoria [8], que consiste en la comparación entre muchos subconjuntos aleatorios, para seleccionar el que menor error de resubstitución tenga. Las variantes evolutivas mas utilizadas son los algoritmos genéticos [92] y el Random Mutation Hill Climbing [64].

Todos estos esquemas dependen muy fuertemente del azar, y solo son satisfactorios para la selección de un número pequeño de objetos en una base de datos pequeña.

5.4.1.9 Basados en proyecciones

Estos esquemas, enunciados inicialmente por Aguilar [75], deciden la pertenencia de un objeto al conjunto resultante basado en sus proyecciones para cada atributo.

5.4.1.10 Conjuntos compactos

El primer reporte de métodos de búsqueda de objetos basados en conjuntos compactos fue realizado por García-Borroto y colaboradores en 2005 [44]. Estos métodos basan su funcionamiento en los conjuntos compactos y β_0-compactos [70], y utilizan la estructura topológica de los datos.

5.4.2 Construcción de objetos

La construcción de objetos, no se limita a escoger un subconjunto de la matriz de entrenamiento, sino que se pueden construir nuevos objetos, si éstos representan a sus respectivas clases de una manera más eficaz. En algunos casos estos algoritmos realizan el proceso de selección de objetos almacenando el promedio de los valores de los rasgos de otros objetos. [64]

Esta estrategia tiene varios puntos débiles:

- Los nuevos objetos construidos pueden ser imposibles de obtener en el dominio real del problema.
- Si en el espacio de representación de los objetos no tiene sentido definir una suma y un producto por un escalar, puede resultar muy complicado definir los conceptos de promedio y de desplazamiento de los valores de los rasgos de un objeto. Esto los hace muy difíciles de utilizar para datos mezclados e incompletos.

5.4.2.1 Mezcla de objetos

Chang [77] fue el primer autor en proponer un esquema de mezcla de objetos cercanos de la misma clase, si esta mezcla no afectaba la eficacia general del clasificador. Muchas modificaciones y mejoras han sido adicionadas posteriormente a este esquema [78, 79], tratando de eliminar los problemas del esquema original.

Estos métodos pueden alterar significativamente las fronteras de decisión por clases, obteniéndose objetos alejados de todos los originales de su misma clase.

5.4.2.2 Aprendizaje competitivo

Esta familia de métodos es capaz de ajustar los objetos de forma tal que la frontera de decisión para cada par de clases vecinas satisfaga aproximadamente la distribución de las clases originales. Pueden encontrarse esquemas postsupervisados (VQ) y presupervisados (LVQ y variantes), pero todos tienen en común el proceso iterativo hasta que se satisfaga cierta condición de estabilidad entre dos iteraciones sucesivas o se alcance un máximo de iteraciones. En cada iteración los objetos (que se seleccionan inicialmente al azar) son movidos para acercarse o alejarse de los objetos originales, según cierta política.

5.4.2.3 Métodos basados en Bootstrap

Esta familia de métodos son variaciones de los métodos de selección aleatoria descritos anteriormente. Se basan en la creación de nuevos objetos promedios de un conjunto de objetos de la misma clase que están cercanos entre sí.

5.4.3 Sinergia de métodos

En la actualidad existe una cantidad enorme, y cada vez mayor de métodos de selección de objetos, cada uno de ellos con sus fortalezas y sus debilidades, cada uno con sus metas y su forma de operar. Los métodos de edición basada en el error están sustentados en una amplia base estadística, pero la mayor parte de los métodos de condensación tiene un basamento heurístico.

Es por esto que se han realizado una gran cantidad de intentos de fusionar estos esquemas para obtener sinergias que sean superiores a cada uno de sus componentes. Un estudio comparativo entre varios esquemas puede ser encontrado en [16]

6 Clasificación taxonómica de los métodos

Tabla 1. Clasificación taxonómica de los métodos

	Método	T1	T2	T3	T4
Selección de Objetos	ENN	np-auto	Presupervisado	Edición	Edición basada en el error
	Allk-NN	np-auto	Presupervisado	Edición	Edición basada en el error
	Multiedit	np-auto	Presupervisado	Edición	Edición basada en el error
	Random Search	np-usuario	Presupervisado	Condensación	Esquemas aleatorios y evolutivos
	RMHC	np-usuario	Presupervisado	Condensación	Esquemas aleatorios y evolutivos
	IB-2	np-auto	Presupervisado	Condensación	Selección guiada por la consistencia
	Shrink	np-auto	Presupervisado	Condensación	Selección guiada por la consistencia
	CNN	np-auto	Presupervisado	Condensación	Selección guiada por consistencia
	RNN	np-auto	Presupervisado	Condensación	Selección guiada por la consistencia
	Métodos basados en gafos de proximidad	np-auto	Presupervisado	Condensación	Grafos de Proximidad
	EOP	np-auto	Presupervisado	Condensación	Basados en proyecciones
	DROP-3	np-auto	Presupervisado	Condensación	Selección guiada por la consistencia
	TIBL	np-auto	Presupervisado	Condensación	Almacenado de instancias típicas
	MCS	np-auto	Presupervisado	Condensación	Selección guiada por consistencia
	CSE	np-auto	Presupervisado	Condensación	Conjuntos compactos
Construcción	Clustering & Relabeling	-	Postsupervisado	Condensación	-
	Bootstrap 4	np-usuario	Presupervisado	Condensación	Esquemas aleatorios y evolutivos
	PNN	np-auto	Presupervisado	Condensación	Mezcla de objetos
	LVQ-1	np-usuario	Presupervisado	Condensación	Aprendizaje Competitivo
	DSM	np-usuario	Presupervisado	Condensación	Aprendizaje Competitivo
	LVQTC	np-adapt	Postsupervisado	Condensación	Aprendizaje Competitivo
	VQ	np-usuario	Postsupervisado	Condensación	Aprendizaje Competitivo

7 Resultados experimentales

En esta sección se exponen los resultados experimentales de la aplicación de los métodos de edición expuestos a un conjunto de bases de datos de la UCI [23]. En la Tabla 2 se ofrece una descripción de las bases de datos escogidas.

Tabla 2. Descripción de las bases de datos

Base de Datos	Cantidad de objetos	Rasgos numéricos	Rasgos no numéricos	Valores perdidos
Glass	214	9	0	0
Pima-indians-diabetes	768	8	0	0
Vehicle	946	18	0	0
Horse-colic	368	8	20	110
Heart-disease-hungarian	294	5	9	779
Heart-disesase-cleveland	303	5	9	6
Annealing	798	9	29	19005
Audiology	226	0	69	291
Auto-mpg	398	8	0	6
Autos	205	16	10	59
Breast-cancer-wpbc	198	32	2	4
Breast-cancer-winconsin	699	0	10	16
Credit-screening	690	6	9	67
Heart-disease-cleveland	303	6	8	0
Herat-disease-hungarian	294	6	8	558
Heart-disease-long-beach	123	6	8	81
Heart-disease-switzerland	200	6	8	273
Hepatitis	115	6	13	167
Horse-colic	368	6	22	1937
Housing	506	12	1	0
Labor-negotiations-C4.5	57	8	8	0
Molecular-biology-promoter-gene-sequences	106	0	57	0
Postoperative-patient-data	90	0	8	3

En las figuras se representa la aplicación de los métodos de selección y/o construcción de objetos a las bases de datos. En el eje x se muestra el porciento de retención y en el eje y el error de clasificación, calculado como cantidad de objetos mal clasificados entre el total de objetos.

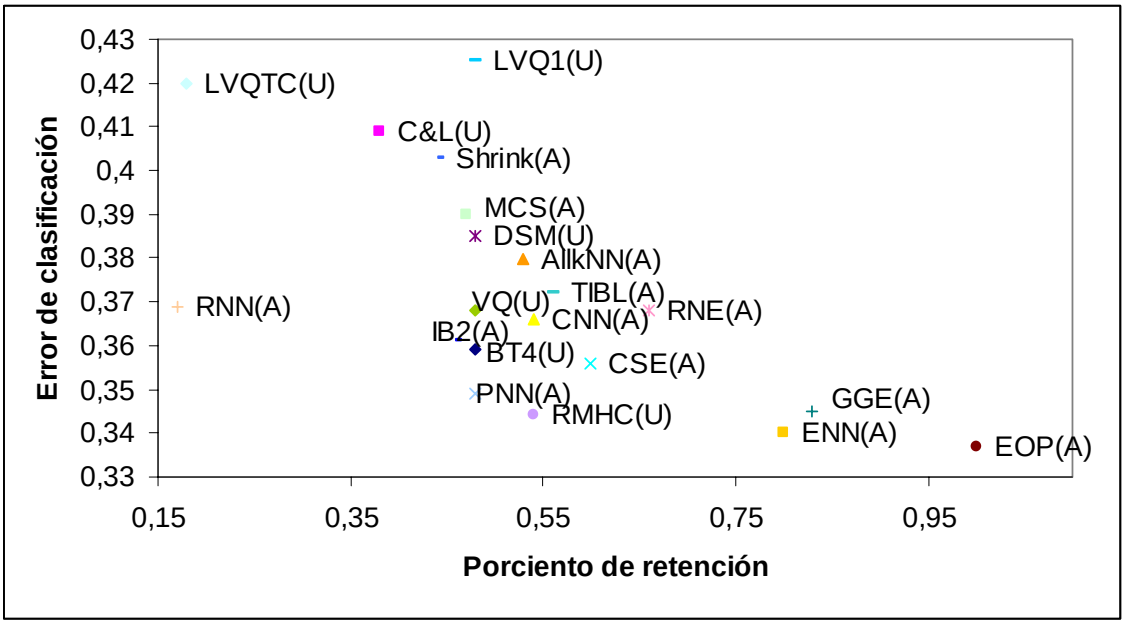

Fig. 38 Resultados de los métodos para la base de datos "Glass".

En esta base de datos, la mayoría de los métodos retienen entre el 50 y el 70% de los objetos. El error se mantiene entre el 30 y el 40%. No se observa superioridad de los métodos que seleccionan los objetos a retener de forma automática con respecto a los métodos donde el número de objetos es definido por el usuario. Métodos como el ENN, el GGC y el EOP logran un error muy bajo, pero con altos niveles de retención, aunque el EOP no logra eliminar ningún objeto. Nótese que hay métodos que logran un bajo porciento de retención, pero con una degradación del clasificador, por ejemplo el MCS y el DSM. Varios métodos logran un buen compromiso entre la retención y el error, entre ellos el IB-2, BT4, CSE, PNN, RMHC, VQ y CNN.

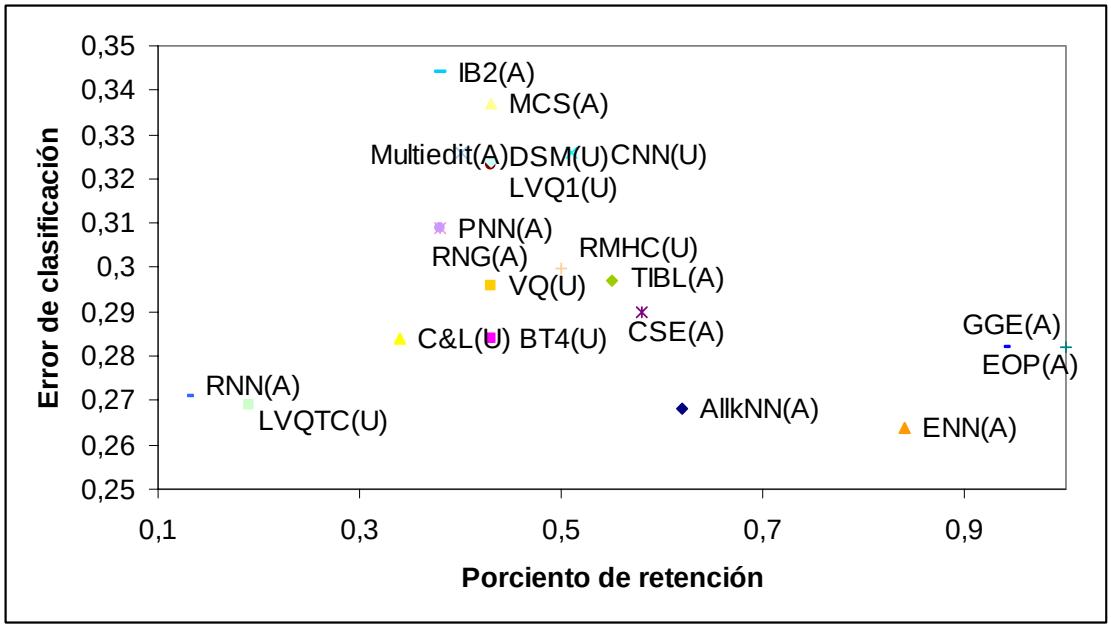

Fig. 39 Resultados de los métodos para la base de datos "Pima-indians-diabetes".

Para la base de datos "Pima-indians-diabetes", la mayoría de los métodos retienen entre el 30 y el 70% de los objetos, mostrando una mayor dispersión que en la base de datos "Glass". El error se mantiene entre el 25 y el 35%. No se observa superioridad de los métodos que seleccionan los objetos a retener de forma automática con respecto a los métodos donde el número de objetos es definido por el usuario. Métodos como el ENN, el GGC y el EOP logran un error muy bajo, pero con altos niveles de retención, aunque el EOP no logra eliminar ningún objeto. Nótese que hay métodos que logran un bajo porciento de retención, pero con una degradación del clasificador, por ejemplo el IB-2 y el MCS. El EOP no logra eliminar ningún objeto. Varios métodos logran un buen compromiso entre la retención y el error, entre ellos el CSE, PNN, RMHC, VQ, TIBL, BT4 y C&R. El RNN y el LVQTC muestran un excelente desempeño en esta base de datos, pues mantienen el error bajo, aun con pocos objetos.

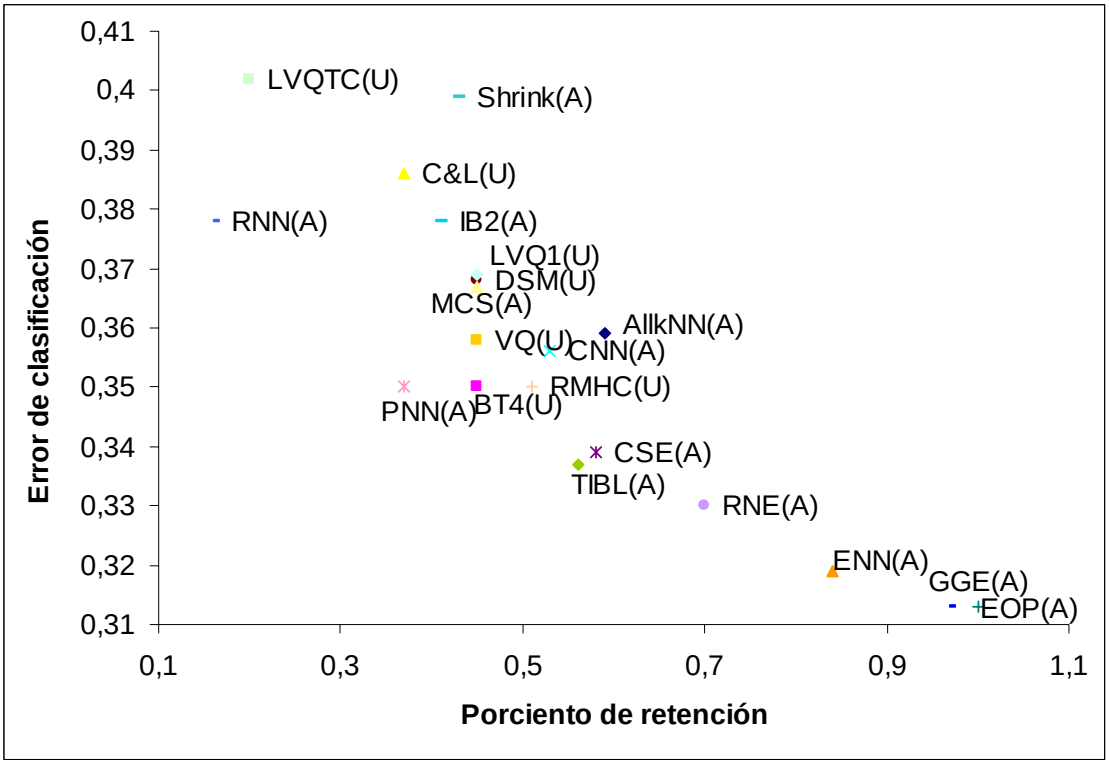

Fig. 40 Resultados para la base de datos "Vehicle".

En esta base de datos, la mayoría de los métodos retienen entre el 50 y el 70% de los objetos. El error se mantiene entre el 30 y el 40%. No se observa superioridad de los métodos que seleccionan los objetos a retener de forma automática con respecto a los métodos donde el número de objetos es definido por el usuario. Métodos como el ENN, el GGC y el EOP logran un error muy bajo, pero con altos niveles de retención, aunque el EOP no logra eliminar ningún objeto. Nótese que hay métodos que logran un bajo porciento de retención, pero con una degradación del clasificador, por ejemplo el LVQTC y el RNN. El EOP no logra eliminar ningún objeto. Varios métodos logran un buen compromiso entre la retención y el error, aunque de forma más dispersa que en otras bases de datos.

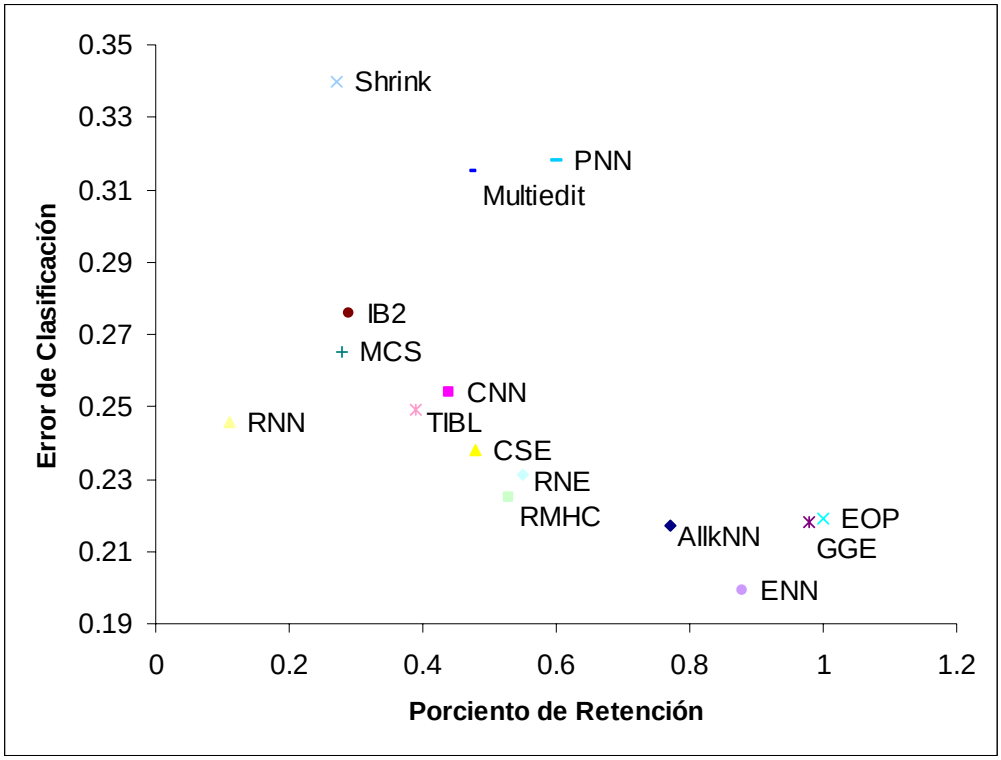

Fig. 41 Resultados de los métodos para la base de datos "Horse-colic".

En esta base de datos, la mayoría de los métodos retienen entre el 20 y el 60% de los objetos. El error de clasificación se mantiene entre el 20 y el 28%. Métodos como el ENN, Allk-NN, el GGE y el EOP logran los niveles de error más bajos, pero con altos niveles de retención; y en el caso del EOP, no logra eliminar ningún objeto. Varios métodos logran un buen compromiso entre la retención y el error, como son RNN, CNN, TIBL, CSE, RNE, RMHC. El RNN se destaca por obtener la menor retención del número de objetos (inferior al 20%) con un error de clasificación cercano (en menos de 5%) al mínimo logrado entre los demás métodos.

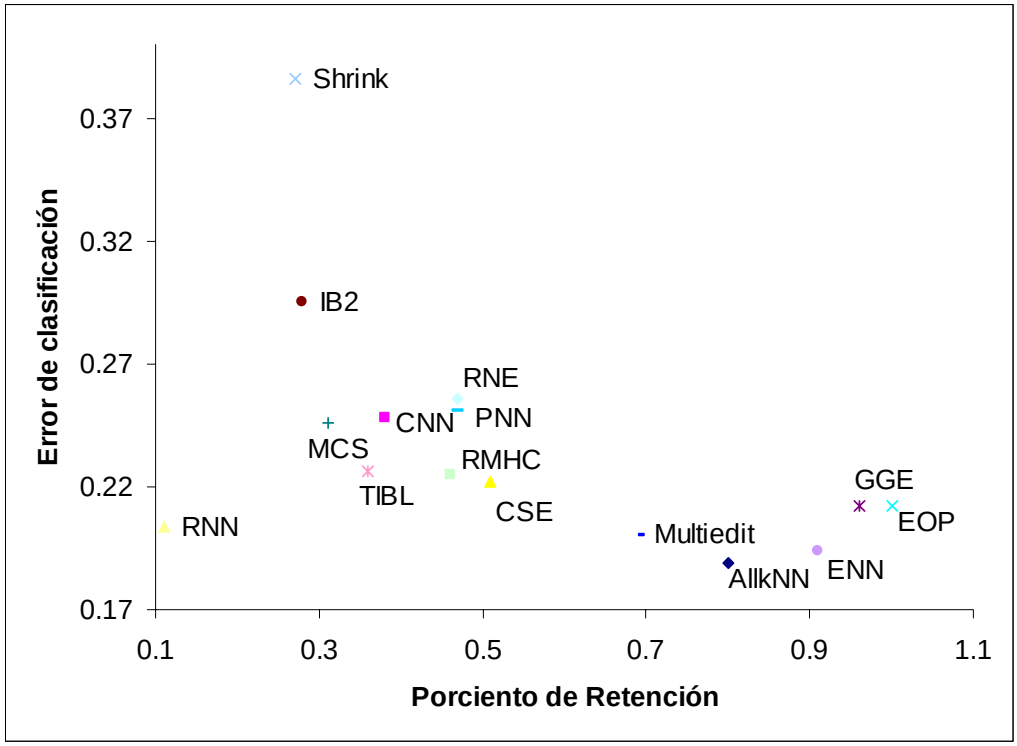

Fig. 42 Resultados de los métodos para la base de datos "Heart-disease-hungarian".

Nótese que la mayoría de los métodos retienen entre el 30 y el 55% de los objetos. El error de clasificación de la mayoría de los métodos es inferior al 27%. Métodos como el ENN, Allk-NN, el GGC y el EOP logran un error muy bajo, pero con altos niveles de retención; y en el caso del EOP, no logra eliminar ningún objeto. Se observa una similitud en los resultados del MCS, TIBL, CNN, RMHC, PNN, CNN, CSE, RNE; estos logran un buen compromiso entre la retención y el error. El caso particular del RNN es de señalar pues este logra eliminar una cantidad significativa de objetos con niveles muy bajos de degradación del clasificador.

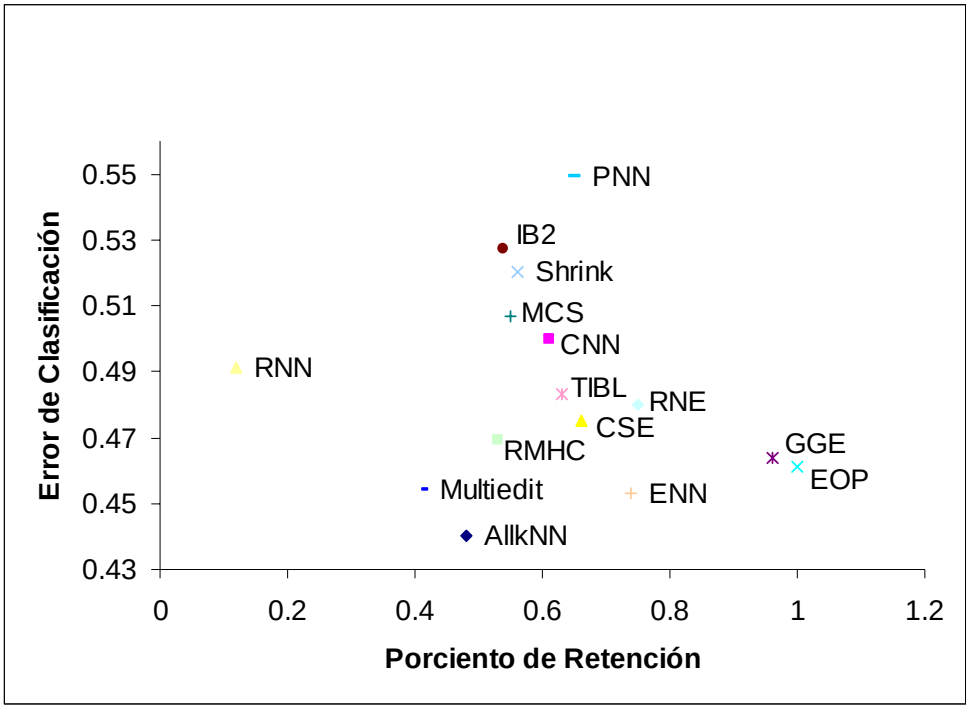

Fig. 43 Resultados de los métodos para la base de datos "Heart-disease-cleveland".

En esta base de datos, la mayoría de los métodos retienen entre el 40 y el 80% de los objetos. El error de clasificación es elevado para todos los métodos (superior al 43%). El RNN se destaca por obtener la menor retención del número de objetos (inferior al 15%) con un error de clasificación cercano (en menos de 6%) al mínimo logrado entre los demás métodos.

8 Conclusiones

En este trabajo se exponen métodos de selección y construcción de objetos para una muestra de entrenamiento en un problema de clasificación supervisada. Se seleccionaron los métodos clásicos, los más referenciados en la literatura y algunos de los más recientes. Los métodos se exponen con una notación uniforme, desambiguando aspectos que no están totalmente claros en las referencias originales.

Un análisis crítico de las características, ventajas y desventajas de cada método es agregado, incluyendo valoraciones reportadas en la literatura, así como conclusiones tomadas de las experimentaciones realizadas. Algunas de estas valoraciones son mostradas en ejemplos sencillos, incluidos en la exposición.

En este reporte se incluyen varias taxonomías, basadas en diferentes criterios, lo que permite un análisis comparativo de los métodos. Una extensa experimentación numérica fue realizada, incluyendo bases de datos con datos mezclados e incompletos y numéricos. Los resultados evidencian las propiedades de cada método, y permiten mostrar su comportamiento ante bases de datos con diferentes propiedades.

Como puede apreciarse casi todos los métodos de selección y construcción de objetos, hasta la fecha, asumen como clasificador la regla del vecino más cercano, a veces extendido a la regla del vecino más similar para datos mezclados e incompletos y disimilaridades no métricas (que no son distancias).

Basándonos en el estudio y las experimentaciones numéricas realizadas podemos enunciar algunos problemas que ameritan nuestra atención futura:

1. Métodos incrementales. Estos permiten trabajar con grandes bases de datos dinámicas, en las que los objetos son agregados y eliminados constantemente. Son capaces de permitir dichas modificaciones sin necesidades de ejecutar todo el algoritmo desde el inicio.
2. Métodos diseñados especialmente para otros clasificadores. Aunque hay reportes del uso de uno de estos métodos originalmente diseñados para el k-NN en otros clasificadores, hay razones lógicas de peso y algunas evidencias experimentales de que esta no es la mejor solución. Un método de selección que tome en cuenta la lógica de funcionamiento particular del clasificador puede brindar resultados superiores.
3. Sinergia de métodos. Aprovechar las ventajas de varios métodos trabajando colectivamente, para suplir las deficiencias de cada uno.
4. Selección asistida del mejor método de edición particular a un problema, lo que puede lograrse en base al cálculo de los metadatos asociados.
5. Selección simultanea de objetos y rasgos, lo que puede brindar resultados superiores a realizar ambas tareas secuencialmente. Si se realizan en cierto orden, la segunda tarea no tiene acceso a toda la información del problema, sino al nuevo problema resultado de la aplicación de la primera.
6. Métodos de selección basados en lógica difusa, y métodos de selección para clasificadores difusos.

Referencias

1. Cover, T.M., Hart, P.E.: Nearest Neighbor pattern classification. IEEE Transactions on Information Theory IT-13, pp. 21-27, (1967)
2. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification (2nd Edition). Wiley-Inter Science (2000)
3. Bezdek, J.C., Kuncheva, L.I.: Nearest prototype classifiers design: an experimental study. International Journal of Intelligent Systems 16, pp. 1445-1473, (2001)
4. Kim, S.W., Oommen, J.B.: A brief taxonomy and ranking of creative prototype reduction schemes. Pattern Analysis and Applications 6**,** pp. 232-244, (2003)
5. Dasarathy, B.D.: Nearest Neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, Los Alamitos, California (1991)
6. Devijver, P.A., Kittler, and J.: Pattern recognition: a statistical approach. Prentice Hall International, Englewood Cliffs, N.J. (1982)
7. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory IT-14**,** pp. 515-516, (1968)
8. Kuncheva, L.I., Bezdek, J.C.: Nearest prototype classification: clustering, genetic algorithms or random search. IEEE Transactions on Systems, Man and Cybernetics SMC-28**,** pp. 160-164, (1998)
9. Gates, G.W.: The reduced nearest neighbor rule. IEEE Transactions on Information Theory IT-18**,** pp. 431-433, (1972)
10. Dasarathy, B.D.: Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design. IEEE Transactions on Systems, Man and Cybernetics SMC-24**,** pp. 511-517, (1994)
11. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man and Cybernetics SMC-2**,** pp. 408-421, (1972)

12. Tomek, I.: An experiment with the Edited Nearest-Neighbor rule. IEEE Transactions on Systems, Man and Cybernetics SMC-6**,** pp. 448-452, (1976)
13. Sanchez, J.S., Barandela, R., Marques, A.I., Alejo, R., Badenas, J.: Analysis of new techniques to obtain quality training sets. Pattern Recognition Letters 24**,** pp. 1015-1022, (2003)
14. Wang, C., Chen, Y.Q.: Improving Nearest Neighbor classification with simulated gravitational collapse. Advances in Natural Computation, PT 3, Proceedings 3612**,** pp. 845-854, (2005)
15. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery 6**,** pp. 153-172, (2002)
16. Dasarathy, B.V., Sanchez, J.S., Townsend, S.: Nearest Neighbor editing and condensing tools - Synergy exploitation. Pattern Analysis and Applications 3**,** pp. 19-30, (2000)
17. Devijver, P.A., Kittler, J.: On the edited nearest neighbor rule. Proceedings of the 5th International Conference on Pattern Recognition, Los Alamitos, California**,** pp. 72-80, (1980)
18. Charu, C.A.: On the effects of dimensionality reduction on high dimensional similarity search. Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, Santa Barbara, California, United States**,** pp. 256-266, (2001)
19. Kuncheva, L.I.: Editing for the k-nearest neighbors rule by a genetic algorithm. Pattern Recognition Letters 16**,** pp. 809-814, (1995)
20. Sanchez, J.S., Pla, F., Ferri, F.J.: Prototype selection for the nearest neighbor rule through proximity graphs. Pattern Recognition Letters 18**,** pp. 507-513, (1997)
21. Li, Y., Huang, J., Zhang, W., Zhang, X.: New prototype selection rule integrated condensing with editing process for the nearest neighbor rules. Proceedings of the IEEE International Conference on Industrial Technology ICIT 2005**,** pp. 950-954, (2005)
22. Wilson, R.D., Martinez, T.R.: Reduction techniques for Instance-based learning algorithms. Machine Learning 38**,** pp. 257-286, (2000)
23. Merz, C.J., Murphy, P.M.: UCI Repository of Machine Learning Databases. University of California at Irvine. Department of Information and Computer Science, Irvine http://www.ics.uci.edu/~mlearn/MLRepository.html (1998)
24. Gómez-Herrera, J.E.: Prognostic of gas-petroleum in the Cuban ophiolitic rocks association, applying mathematical modeling. Geofísica Internacional 33**,** pp. 447-467, (1994)
25. Martinez Trinidad, J.F., Velasco-Sánchez, M., Contreras-Arévalo, E.E.: Discovering differences in patients with uveitis through typical testors by class. Lecture Notes on Computer Science 1910**,** pp. 524-529, (2000)
26. Ruiz-Shulcloper, J., Abidi, M.A.: Logical Combinatorial Pattern Recognition: A review. In: Pandalai, S.G. (ed.): Recent Research Developments in Pattern Recognition. Transword Research Networks, USA**,** pp. 133-176 (2002)
27. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. Proceedings of the 12th International Conference on Machine Learning**,** pp. 194-202, (1995)
28. Lumijarvi, J., Laurikkala, J., Juhola, M.: A comparison of different heterogeneous proximity functions and Euclidean distance. Medinfo 11**,** pp. 1362-1366, (2004)
29. Ruiz-Shulcloper, J., Pico Peña, R., Ibarra, C.A., Hernández, G.V., Martín, A.M.: Modelación matemática del problema de discriminación de anomalías AGE perspectivas para rocas fosfóricas de génesis sedimentaria. Ciencias Matemáticas 13**,** pp. 159-171, (1992)
30. Friedman, J.H.: An overview of predictive learning and function approximation. In: Cherkassky, V., Friedman, J.H., Wechsler, H. (eds.): From statistics to neural networks: Theory and Pattern Recognition Applications**,** pp. 1-61, (1993)
31. Duch, W., Grudzinski, K., G., S.: Symbolic features in neural networks. Proceedings of the 5th Conference on Neural Networks and Soft Computing, Zakopane**,** pp. 180-185, (2000)
32. Stanfill, C., Walts, D.L.: Toward memory-based reasoning. Communications of the ACM 29**,** pp. 1213-1228, (1986)
33. Maudes, J., Rodríguez, J.J., García-Osorio, C.: Cascading for nominal data. Lecture Notes in Computer Science 4472**,** pp. 231-240, (2007)
34. Wilson, R.D., Martinez, T.R.: Improved heterogeneous distance functions. Journal of Artificial Intelligence Research 6**,** pp. 1-34, (1997)
35. Pekalska, E., Paclík, P., Duin, R.P.W.: A generalized kernel approach to dissimilarity-based classification. Journal of Machine Learning Research 2**,** pp. 175-211, (2001)
36. Pekalska, E., Duin, R.P.W.: Automatic pattern recognition by similarity representations. IEEE Electronic Letters 32**,** pp. 159-160, (2001)

37. Martínez-Trinidad, J.F., Guzmán-Arenas, A.: The logical combinatorial approach to pattern recognition: an overview through selected works. Pattern Recognition 34**,** pp. 741-751, (2001)
38. Ruiz-Shulcloper, J., Guzmán-Arenas, A., Martinez Trinidad, J.F.: Logical combinatorial approach to pattern recognition: Feature selection and supervised classification. Editorial Politécnica, Mexico (2000)
39. Little, R.J.A., Rubin, D.B.: Statistical analysis with missing data. Wiley, New York (2002)
40. Schafer, J.L., Graham, J.W.: Missing data: our view of the state of the art. Psychological Methods 7**,** pp. 147-177, (2002)
41. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning 6**,** pp. 37-66, (1991)
42. Ventura, D., Martínez, T.R.: An empirical comparison of discretization methods. Proceedings of the 10th International Symposium on Computer and Information Sciences**,** pp. 443-450, (1995)
43. Toussaint, G.T.: Proximity graphs for nearest neighbor decision rules: Recent progress. Proceedings of the 34th Symposium on Computing and Statistics INTERFACE-2002, Montreal, Canada**,** pp. 1-20, (2002)
44. García-Borroto, M., Ruiz-Shulcloper, J.: Selecting prototypes in mixed incomplete data. Lecture Notes in Computer Science 3773**,** pp. 450-459, (2005)
45. Hattori, K., Takahashi, M.: A new edited k-nearest neighbor rule in the pattern classification problem. Pattern Recognition 33**,** pp. 521-528, (2000)
46. Bezdek, J.C., Kuncheva, L.I.: Nearest Prototype classifiers design: an experimental study. Technical Report 00014-96-1-0642, University of California (2004)
47. Kuncheva, L.I.: Combining pattern classifiers: Methods and algorithms. John Wiley & Sons (2004)
48. Ullmann, J.R.: Automatic selection of reference data for use in a nearest neighbor method of pattern classification. IEEE Transactions on Information Theory IT-20**,** pp. 541-543, (1974)
49. Tomek, I.: Two modifications of CNN. IEEE Transactions on Systems, Man and Cybernetics SMC-6**,** pp. 769-772, (1976)
50. Kibler, D., Aha, D.W.: Learning representative exemplars of concepts: An initial case study. Proceedings of the 4th International workshop on Machine learning, Irvine, CA**,** pp. 24-30, (1987)
51. Angiulli, F.: Fast condensed nearest neighbor rule. Proceedings of the 22nd International conference on Machine learning, Bonn, Germany, ACM International Conference Proceeding Series 119**,** pp. 25-32, (2005)
52. Swonger, C.W.: Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition. In: Watanabe, S. (ed.): Frontiers of Pattern Recognition. Academic Press, New York**,** pp. 511-519, (1972)
53. Ritter, G.L., Woodruff, H.B., Lowry, S.R., Isenhour, T.L.: An algorithm for a selective nearest neighbor decision rule. IEEE Transactions on Information Theory IT-21**,** pp. 665-669, (1975)
54. Zhang, H., Sun, G.: Optimal reference subset selection for nearest neighbor classification by Tabu search. Pattern Recognition 35**,** pp. 1481-1490, (2002)
55. Dasarathy, B.V.: Nearest unlike neighbor (NUN): An aid to decision confidence estimation. Optical Engineering 34**,** pp. 2785-2792, (1995)
56. Wilfong, G.: Nearest neighbor problems. Proceedings of the 7th Annual ACM Symposium on Computational Geometry**,** pp. 224-233, (1991)
57. Jaromczyk, J.W., Toussaint, G.T.: Relative neighborhood graphs and their relatives. Proceedings of IEEE 80(9)**,** pp. 1502-1517, (1992)
58. Bhattacharya, B.K., Poulsen, R.S., Toussaint, G.T.: Application of proximity graphs to editing nearest neighbor decision rules. Technical Report SOCS 92.19, School of Computer Science, McGill University, Montreal (1992)
59. Delaunay, B.: Sur la sphère vide. Bulletin Academy Science USSR 7**,** pp. 793-800, (1934)
60. Takekazu, K., Toshikazu, W.: Direct condensing: An efficient Voronoi condensing algorithm for nearest neighbor classifiers. Pattern Recognition 3**,** pp. 474- 477, (2004)
61. Barber, C.B., David, P.D., Hannu, H.: The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software 22**,** pp. 469-483, (1996)
62. Kuncheva, L.I.: Combining pattern classifiers: Methods and algorithms. Wiley-Inter Science (2004)
63. Chang, I.E., Lippman, R.P.: Using genetic algorithms to improve pattern classification performance. Advances in Neural Information Processing 3**,** pp. 797-803, (1991)
64. Skalak, D.B.: Prototype and feature selection by sampling and random mutation hill climbing algorithms. Proceedings of the 11th International Conference on Machine Learning, New Brunswick**,** pp. 293-301, (1994)
65. Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform Hill-Climbing? In Cowan, J.D., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems 6 Morgan Kaufmann (1994)

66. Zhang, J.: Selecting typical instances in Instance-Based learning. Proceedings of the 9th International Machine Learning Workshop, Aberdeen, Scotland 470-479 (1992)
67. Wilson, R.D.: Advances in Instance-Based learning algorithms. PhD. Thesis, Department of Computer Science of Brigham Young University, Brigham (1997)
68. Fred, G., Fred, L.: Tabu Search. Kluwer Academic Publishers (1997)
69. Huang, D., Chow, T.W.S.: Enhancing density-based data reduction using entropy. Neural Computation 18 470-495 (2006)
70. Martínez-Trinidad, J.F., Ruiz-Shulcloper, J., Lazo-Cortés, M.S.: Structuralization of universes. Fuzzy Sets and Systems 112 485-500 (2000)
71. Li, Y.G., Hu, Z.H., Cai, Y.Z., Zhang, W.D.: Support vector based prototype selection method for Nearest Neighbor rules. Advances in Natural Computation, PT 1, Proceedings 3610 528-535 (2005)
72. Huang, C.C.: A novel gray-based reduced NN classification method. Pattern Recognition 39 1979-1986 (2006)
73. Deng, J.L.: Introduction to Grey system theory. Grey Systems 1 1-24 (1989)
74. Fayed, H.A., Hashem, S.R., Atiya, A.F.: Self-generating prototypes for pattern classification. Pattern Recognition 40 1498-1509 (2007)
75. Aguilar, J.S., Riquelme, J.C., Toro, M.: Data set editing by ordered projection. Intelligent Data Analysis 5(5) 405-417 (2001)
76. Davison, A.C., Hinkley, D.V.: Bootstrap methods and their application. Cambridge Press (1997)
77. Chang, C.L.: Finding prototypes for nearest neighbor classifier. IEEE Transactions on Computers C-23 1179-1184 (1974)
78. Mollineda, R.A., Ferri, F.J., Vidal, E.: An efficient prototype merging strategy for the condensed 1-NN rule through class conditional hierarchical clustering. Pattern Recognition 35 2771-2782 (2002)
79. Bezdek, J.C., Reichherzer, T.R., Lim, G.S., Y., A.: Multiple-prototype classifier design. IEEE Transactions on Systems, Man and Cybernetics SMC-28 67-79 (1998)
80. Kohonen, T.: Self - Organizing maps. Springer, Germany (1995)
81. Geva, S., Sitte, J.: Adaptive nearest neighbor pattern classification. IEEE Transactions on Neural Networks 2 318-322 (1991)
82. Odorico, R.: Learning vector quantization with training counters (LVQTC). Neural Networks 10 1083-1088 (1997)
83. Xie, Q., Laszlo, C.A., Ward, R.K.: Vector quantization technique for nonparametric classifier design. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-15 1326-1330 (1993)
84. Hamamoto, Y., Uchimura, S., Tomita, S.: A bootstrap technique for Nearest Neighbor classifier design. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-19 73-79 (1997)
85. Mollineda, R.A., Ferri, F.J., Vidal, E.: Merge-based prototype selection for Nearest Neighbor classification. Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics 7 640-645 (2000)
86. Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., Torkkola, K.: LVQ_PAK: The Learning Vector Quantization program package. Technical Report A-30, Faculty of Information Technology, Helsinki University of Technology, Finland (1996)
87. Kuncheva, L.I., Bezdek, J.C.: An integrated framework for generalized nearest prototype classifier design. International Journal of Uncertainty, Fuzziness and Knowledge-based systems 6 437-457 (1998)
88. Paredes, R., Vidal, E.: Weighting prototypes: A new editing approach. Proceedings of the 15th International Conference on Pattern Recognition (ICPR'00) 2 2025 (2000)
89. Sklansky, J., Michelotti, L.: Locally trained piecewise linear classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2 101-111 (1980)
90. Priebe, C.E., DeVinney, J.G., Marchette, D.J.: On the distribution of the domination number for random class cover catch digraphs. Statistics and Probability Letters 55 239-246 (2001)
91. Priebe, C.E., Marchette, D.J., DeVinney, J.G., Socolinsky, D.: Classification using class cover catch digraphs. PhD. Thesis, Johns Hopkins University, Barlimore (2002)
92. Kuncheva, L.I.: Fitness functions in editing k-NN reference set by genetic algorithms. Pattern Recognition 30, No. 6 1041-1049, (1997)

Anexos

A continuación, para cada base de datos se muestran los diferentes métodos. De cada método se muestran las diferencia entre la eficacia del clasificador utilizando toda la matriz de aprendizaje y el resultado de la edición (Δ), el porciento de compactación (% Red.) y el tiempo (en segundos) de la ejecución. La diferencia de eficacia es negativa cuando ocurre un aumento de los errores de clasificación. El tiempo de ejecución de cada método es dependiente de la estación donde se corrieron las experimentaciones, y sólo debe ser utilizado para las comparaciones entre métodos.

Tabla 3. Resultados de "Glass"

Método	Δ	**% Red.**	**T(sec)**
Allk-NN	-4.28	47.4	1.81
BT4	-2.21	51.67	0.03
C&L	-7.18	62.08	0.04
CNN	-2.87	46.47	0.52
CSE	-1.9	40.33	0.2
DSM	-4.83	51.67	8.99
EOP	0	0	0.01
GGE	-0.77	17.47	1.45
IB2	-2.45	54.09	0.05
LVQ1	-8.81	51.67	39.83
LVQTC	-8.34	81.6	8.76
MCS	-5.29	52.79	1.04
Multiedit	-24.9	73.05	0.1
PNN	-1.22	51.86	28.33
RNE	-3.06	34.39	1.21
RMHC	-0.65	45.54	4.47
RNN	-3.25	83.09	14.11
Shrink	-6.63	55.58	0.14
TIBL	-3.5	44.24	1.53
VQ	-3.07	51.67	0.16
ENN	-0.33	20.07	0.08

Tabla 4. Resultados de "Pima-indians-diabetes"

Método	Δ	**% Red.**	**T(sec)**
Allk-NN	1.4	37.57	23.41
BT4	-0.21	56.77	2.12
C&L	-0.24	65.67	0.44
CNN	-4.41	48.74	1.69
CSE	-0.85	42.31	1.33
DSM	-4.16	56.77	146.39
EOP	0	0	0.02
GGE	0	5.87	16.61
IB2	-6.24	62.22	0.16
LVQ1	-4.29	56.77	220.34
LVQTC	1.21	81.11	46.95
MCS	-5.5	57.08	5.42
Multiedit	-4.47	60.01	1.39
PNN	-2.69	62.33	558.23
RNE	-2.58	38.09	11.18
RMHC	-1.8	49.56	74.58
RNN	1.11	87.08	373.74
Shrink	-9.73	59.5	0.65
TIBL	-1.55	44.67	103.9
VQ	-1.46	56.77	2.56
ENN	1.76	15.54	0.92

Tabla 5. Resultados de "Vehicle"

Método	Δ	% Red.	T(sec)
Allk-NN	-4.63	41.12	82.21
BT4	-3.75	55.37	1.53
C&L	-7.27	63.09	2.85
CNN	-4.29	47.13	4.19
CSE	-2.61	42.09	9.35
DSM	-5.46	55.37	268.09
EOP	0	0	0.01
GGE	0	3.28	32.21
IB2	-6.5	59.07	0.48
LVQ1	-5.58	55.37	257
LVQTC	-8.92	79.97	122.58
MCS	-5.39	55.18	22.02
Multiedit	-28.2	78.17	3.02
PNN	-3.71	63.04	1476.3
RNE	-1.72	29.88	19.91
RMHC	-3.66	48.57	148.6
RNN	-6.49	83.53	1212.59
Shrink	-8.63	56.71	1.42
TIBL	-2.46	44.03	565.38
VQ	-4.53	55.37	13.36
ENN	-0.59	15.54	9.82

Tabla 6. Resultados de "Horse-colic"

Método	Δ	% Red.	T(sec)
Allk-NN	0.13	23.35	3.15
CNN	-3.56	55.74	0.57
CSE	-1.99	51.98	0.3
EOP	-0.07	0	0.02
GGE	0	2.45	15.12
IB2	-5.78	71	0.08
MCS	-4.65	71.94	1.13
Multiedit	-9.65	52.92	0.67
PNN	-9.93	39.74	108.37
RNE	-1.29	44.63	7.31
RMHC	-0.72	46.7	33.39
RNN	-2.81	88.89	43.24
Shrink	-12.3	73.45	0.21
TIBL	-3.1	61.21	6.88
ENN	1.91	12.05	0.72

Tabla 7. Resultados de "Heart-disease-hungarian"

Método	Δ	% Red.	T(sec)
Allk-NN	2.25	20.48	1.15
CNN	-3.61	61.67	0.12
CSE	-1.09	49.12	0.14
EOP	0	0	0.01
GGE	-0.1	3.74	4.61
IB2	-8.31	71.81	0.03
MCS	-3.49	69.16	0.43
Multiedit	1.18	31.5	0.43
PNN	-3.95	52.86	29.65
RNE	-4.44	52.86	2.03
RMHC	-1.4	53.52	9.39
RNN	0.79	89.21	14.07
Shrink	-17.4	72.91	0.1
TIBL	-1.42	63.88	3.08
ENN	1.75	9.03	0.16

Tabla 8. Resultados de "Heart-disease-cleveland"

Método	Δ	% Red.	T(sec)
Allk-NN	2.08	52.23	2.43
CNN	-3.88	38.64	0.28
CSE	-1.42	34.18	0.17
EOP	0	0	0.02
GGE	-0.28	4.46	2.03
IB2	-6.6	46.07	0.14
MCS	-4.59	44.59	1.49
Multiedit	0.71	58.6	0.17
PNN	-8.74	35.03	41.35
RNE	-1.92	25.05	1.57
RMHC	-0.82	47.13	12.53
RNN	-3.01	88.32	17.84
Shrink	-5.91	43.52	0.22
TIBL	-2.21	37.15	3.6
ENN	0.82	25.9	0.22

Tabla 9. Resultados de "Annealing"

Método	Δ	**% Red.**	**T(sec)**
Allk-NN	-5.48	16.64	29.52
CNN	1.99	81.49	0.63
CSE	-0.04	55.29	1.43
EOP	-0.11	0	0.04
IB2	-0.72	83.2	0.55
MCS	0.79	84.84	6.37
Multiedit	-10.6	26.91	6.04
RNE	-0.68	65.79	24.05
RNN	-2.89	90.12	576.84
Shrink	-11.3	86.24	3.13
ENN	-2.44	5.29	2.62

Tabla 10. Resultados de "Audiology"

Método	Δ	**% Red.**	**T(sec)**
Allk-NN	-18.89	52.34	6.2
CNN	-1.16	45.79	0.55
CSE	-0.39	35.83	0.47
EOP	-0.87	0	0.06
IB2	-0.62	46.11	0.22
MCS	-1.26	41.43	3.78
Multiedit	-32.1	81.93	0.84
RNE	0	3.43	2.43
RMHC	-6.08	42.68	30.37
RNN	-10.4	81	37.27
Shrink	-1.74	51.71	0.58
TIBL	-0.46	47.35	6.8
ENN	-9.23	20.25	0.4

Tabla 11. Resultados de "Auto-mpg"

Método	Δ	**% Red.**	**T(sec)**
Allk-NN	-1.87	95.5	1.45
CNN	0	1.24	0.26
CSE	0	1.55	0.15
EOP	0	0	0
GGE	0	0.16	3.14
IB2	0	1.86	0.08
MCS	0	1.55	1.65
Multiedit	-2.26	98.91	0.05
RNE	0	0.31	0.99
RMHC	-1.03	46.27	13.3
RNN	-1.33	96.27	14.73
Shrink	0	1.55	0.12
TIBL	0	1.4	8.02
ENN	-1.48	64.13	0.16

Tabla 12. Resultados de "Autos"

Método	Δ	**% Red.**	**T(sec)**
Allk-NN	-13.13	57.14	1.31
CNN	-0.91	36.21	0.34
CSE	-1	36.54	0.19
EOP	0	0	0.01
IB2	-3.68	41.86	0.07
MCS	-2.48	44.19	1.92
Multiedit	-30.4	93.69	0.07
RNE	-0.99	19.93	1.22
RMHC	-1.97	45.85	8.9
RNN	-9.09	78.74	11.49
Shrink	-3.95	45.85	0.16
TIBL	-2.28	35.88	3.37
ENN	-4.11	21.93	0.14

Tabla 13. Resultados de "Breast-cancer-wpbc"

Método	Δ	% Red.	T(sec)
Allk-NN	12.08	37.68	4.97
CNN	-3.22	47.25	0.44
CSE	-1.77	44.35	0.55
EOP	0	0	0.01
GGE	0.14	4.06	2.57
IB2	-7.71	59.13	0.04
MCS	-5.01	57.68	1.66
Multiedit	12.63	46.09	0.18
PNN	-1.81	62.03	15.3
RNE	-4.12	45.51	4.82
RMHC	1.75	46.38	8.4
RNN	7.65	90.14	41.12
Shrink	-9.25	60.29	0.21
TIBL	-1.59	46.67	2.19
ENN	7.95	15.94	0.18

Tabla 14. Resultados de "Breast-cancer-winconsin"

Método	Δ	% Red.	T(sec)
Allk-NN	0.82	5.3	8.09
CNN	-1.93	89.74	0.28
CSE	-1.15	70.61	0.79
EOP	0.1	0	0.05
GGE	0	0	13.15
IB2	-2.34	91.39	0.05
MCS	-2.86	87.3	1.56
Multiedit	-1.09	15.65	5.82
RNE	-0.25	32.09	29.05
RMHC	-0.18	48.43	55.06
RNN	-0.84	94.78	194.49
Shrink	-9.8	92.78	0.69
TIBL	-0.91	88.96	3.53
ENN	0.3	2.26	1.08

Tabla 15. Resultados de "Credit-screening"

Método	Δ	% Red.	T(sec)
Allk-NN	5.16	24.04	67.61
CNN	-5.65	59.2	3.33
CSE	-2.96	47.8	9.42
EOP	0	0	0.06
IB2	-9.47	69.54	0.91
MCS	-10.2	66.67	18.09
Multiedit	2.06	43.01	13.75
RNE	-0.98	41.57	64.86
RMHC	0.1	50.19	186.9
RNN	0.65	88.7	1447.2
Shrink	-18.0	71.46	3.05
TIBL	-5.76	56.23	94.71
ENN	4.76	10.63	2.97

Tabla 16. Resultados de "Heart-disease-cleveland"

Método	Δ	% Red.	T(sec)
Allk-NN	2.08	52.23	2.43
CNN	-3.88	38.64	0.28
CSE	-1.42	34.18	0.17
EOP	0	0	0.02
GGE	-0.28	4.46	2.03
IB2	-6.6	46.07	0.14
MCS	-4.59	44.59	1.49
Multiedit	0.71	58.6	0.17
PNN	-8.74	35.03	41.35
RNE	-1.92	25.05	1.57
RMHC	-0.82	47.13	12.53
RNN	-3.01	88.32	17.84
Shrink	-5.91	43.52	0.22
TIBL	-2.21	37.15	3.6
ENN	0.82	25.9	0.22

Tabla 17. Resultados de "Herat-disease-hungarian"

Método	Δ	% Red.	T(sec)
Allk-NN	2.25	20.48	1.15
CNN	-3.61	61.67	0.12
CSE	-1.09	49.12	0.14
EOP	0	0	0.01
GGE	-0.1	3.74	4.61
IB2	-8.31	71.81	0.03
MCS	-3.49	69.16	0.43
Multiedit	1.18	31.5	0.43
PNN	-3.95	52.86	29.65
RNE	-4.44	52.86	2.03
RMHC	-1.4	53.52	9.39
RNN	0.79	89.21	14.07
Shrink	-17.4	72.91	0.1
TIBL	-1.42	63.88	3.08
ENN	1.75	9.03	0.16

Tabla 18. Resultados de "Heart-disease-long-beach"

Método	Δ	% Red.	T(sec)
Allk-NN	-2.65	82.91	0.47
CNN	-1.33	12.34	0.05
CSE	-1.17	15.19	0.07
EOP	0	0	0.01
GGE	0	0.63	0.54
IB2	-2.07	21.52	0.03
MCS	-2.91	17.41	0.39
Multiedit	-6.17	92.41	0.03
PNN	-2.92	9.18	4.5
RNE	-0.29	7.91	0.17
RMHC	-2.96	44.94	8.82
RNN	-5.3	86.71	5.23
Shrink	-1.75	17.72	0.07
ENN	1.02	44.3	0.09

Tabla 19. Resultados de "Heart-disease-switzerland"

Método	Δ	% Red.	T(sec)
Allk-NN	-1.95	82.51	0.22
CNN	-0.99	13.66	0.03
CSE	-1.55	16.39	0.03
EOP	0	0	0
GGE	-0.23	1.09	0.23
IB2	-0.65	26.78	0.02
MCS	-0.81	18.03	0.25
Multiedit	3.26	91.26	0.02
PNN	-2.37	9.84	1.96
RNE	0.12	10.38	0.34
RMHC	-4.86	38.25	1.34
RNN	2.43	86.34	0.87
Shrink	-1.32	16.94	0.01
TIBL	0.24	8.2	0.85
ENN	-2.38	44.26	0.03

Tabla 20. Resultados de "Hepatitis"

Método	Δ	% Red.	T(sec)
Allk-NN	-0.48	22.86	1.37
CNN	-6	59.64	0.06
CSE	-1.86	50.71	0.18
EOP	0	0	0.02
GGE	-1.01	29.29	2.52
IB2	-5.75	71.43	0.02
MCS	-8.11	72.5	0.53
Multiedit	-1.49	28.57	0.33
PNN	-9.74	45.36	9.87
RNE	-3.28	61.07	1.19
RMHC	-0.95	43.57	5.82
RNN	1.15	88.57	8.35
Shrink	-11.8	70.71	0.04
ENN	3.07	10	0.09

Tabla 21. Resultados de "Horse-colic"

Método	Δ	% Red.	T(sec)
Allk-NN	0.13	23.35	3.15
CNN	-3.56	55.74	0.57
CSE	-1.99	51.98	0.3
EOP	-0.07	0	0.02
GGE	0	2.45	15.12
IB2	-5.78	71	0.08
MCS	-4.65	71.94	1.13
Multiedit	-9.65	52.92	0.67
PNN	-9.93	39.74	108.37
RNE	-1.29	44.63	7.31
RMHC	-0.72	46.7	33.39
RNN	-2.81	88.89	43.24
Shrink	-12.1	73.45	0.21
TIBL	-3.1	61.21	6.88
ENN	1.91	12.05	0.72

Tabla 22. Resultados de "Housing"

Método	Δ	% Red.	T(sec)
Allk-NN	-1.13	97.3	2.39
CNN	0	0.21	0.24
CSE	0	0.41	0.17
EOP	0	0	0.01
GGE	0	0.21	1.29
IB2	0	0.41	0.13
MCS	0	0.41	2.34
Multiedit	-1.09	98.96	0.06
PNN	0	0.41	2.26
RNE	0	0.21	0.89
RMHC	0	45.23	7.42
RNN	-1.13	98.13	25.52
Shrink	0	0.41	0.17
TIBL	0	0.21	23.18
ENN	-0.3	65.15	0.15

Tabla 23. Resultados de "Labor-negotiations-C4.5"

Método	Δ	% Red.	T(sec)
Allk-NN	-4.74	33.7	0.15
CNN	-1.7	58.7	0.02
CSE	4.41	53.26	0.02
EOP	1.33	0	0
IB2	-15.8	64.13	0
MCS	0.41	75	0.13
Multiedit	-36.4	61.96	0.02
RNE	-4.97	44.57	0.12
RMHC	-2.67	48.91	0.77
RNN	-0.55	85.87	1.09
Shrink	-17.1	70.65	0.01
TIBL	4.41	65.22	0.1
ENN	0	5.43	0.03

Tabla 24. Resultados de "Molecular-biology-promoter-gene-sequences"

Método	Δ	% Red.	T(sec)
Allk-NN	-7.71	29.03	1.13
CNN	-4.01	51.61	0.24
CSE	-0.5	49.68	0.09
EOP	2.92	0	0.02
IB2	-4.33	61.29	0.07
MCS	1.57	62.58	0.78
Multiedit	-24.4	61.94	0.12
RNE	-0.01	7.74	0.74
RMHC	-2.88	39.35	7.67
RNN	-6.74	75.48	4.15
Shrink	-11.8	72.26	0.08
TIBL	2.45	61.29	0.43
ENN	-1.55	9.68	0.13

Tabla 25. Resultados de "Postoperative-patient-data"

Método	Δ Efic	**% Comp**	**T(sec)**
Allk-NN	13.27	40.3	0.12
CNN	-6.18	39.55	0.01
CSE	-1.38	26.87	0.01
EOP	1.28	0	0
IB2	-4.76	51.49	0
MCS	1.59	31.34	0.06
Multiedit	16.04	47.76	0.01
RNE	-0.31	13.43	0.07
RMHC	2.87	38.81	0.66
RNN	12.99	86.57	0.38
Shrink	-8.63	50.75	0.01
ENN	9.17	23.88	0.01

RT_003, Julio 2008
Aprobado por el Consejo Científico CENATAV

Editor: **Lic. Margarita Ilisástigui Avilés**
Diseño de Portada: **DCG Matilde Galindo Sánchez**
RNPS No. 2142
ISSN Solicitado
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. No. 21812 e/218 y 222, Rpto. Siboney, Playa;
Ciudad de La Habana. Cuba. C.P. 12200
Impreso en Cuba