Nuevos Métodos de Selección de Objetos y su Aplicación Sinérgica

por

Milton García Borroto

Tesis sometida como requisito parcial para
obtener el grado de

Maestro en Ciencias en el área de Ciencias Computacionales

en el
Instituto Nacional de Astrofísica,
Óptica y Electrónica

Supervisada por:

Dr. José Francisco Martínez Trinidad
INAOE
Dr. José Ruiz Shulcloper
CENATAV, Cuba

Dedicatoria

A mi hija
A mis padres
A mi familia
A mis amigos

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) y el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), por el soporte económico de mis estancias en México, por tener un Instituto de este nivel, y un programa de Maestría con tanta calidad.
A mis tres padres, sin ellos no sería quien soy.
A toda mi familia, por sacrificar mucho tiempo que debí compartir con ellos, por entenderlo, y hacerlo con alegría.
A todos mis profesores de la Maestría en Ciencias Computacionales del INAOE, de los que me nutrí no sólo docentemente, sino profesionalmente.
A Shul, que me guió en mis primeros pasos por la ciencia, y me sigue soportando hasta ahora
A Pancho y Ariel, sin cuyo apoyo habría sido difícil tener este resultado
A mi comité de revisión: Dr. Jesús González, Dr. Ariel Carrasco y Dr. Carlos Reyes, por los comentarios y sugerencias al contenido y estilo del trabajo. También al Dr. Eduardo Morales, quién sugirió la idea de utilizar la frontera de Pareto.
A todo el personal del INAOE, en especial a Martha Olmos, Rocío Rodas, Conchita Tecuatl y Carmen Meza, que me dieron siempre su ayuda con una sonrisa en los labios.
A mis amigos cubanos, especialmente Leya y Migue, sin cuyo apoyo remoto, la lejanía hubiera sido mucho más dura de llevar.
A mis cuates mexicanos: Alex, Jeza, Pato, Artemio y Beto por hacerme sentir como en mi propia tierra

Resumen

La regla de los k Vecinos más Cercanos es uno de los clasificadores más populares en la actualidad, tanto por su sencillez de funcionamiento y buenos resultados prácticos, como por no necesitar ningún conocimiento *a priori* de las distribuciones de los datos. Sin embargo, para resolver problemas prácticos, presenta varios inconvenientes. En este trabajo se aborda la solución de dos de los más importantes: la intolerancia a objetos ruidosos, y el alto costo computacional de la clasificación.

En la revisión bibliográfica realizada, se pudo verificar que existen una cantidad muy grande de métodos, con comportamientos disimilares en diferentes bases de datos. Otra tendencia detectada es la utilización de combinaciones de métodos, que aprovechen de manera sinérgica las ventajas de cada método, ayudando a mitigar sus puntos débiles.

En este trabajo se presentan dos nuevos métodos de edición y dos nuevos métodos de condensación, con buen comportamiento individual, y muy buen comportamiento en combinaciones, tanto con datos mezclados e incompletos como con datos numéricos. La modificación apreciable de la frontera de Pareto al incluirlos en las comparaciones muestra que se realizó un aporte al estado del arte.

Un estudio con mayor profundidad de los resultados de diferentes estrategias de combinación de métodos permitió enunciar propiedades de cada una de ellas. Además se determinó la dependencia entre la calidad de los resultados de cada estrategia y el entremezclado de las clases, medido mediante un meta – dato.

Abstract

The k Nearest Neighbor rule is one of the most popular classifiers used nowadays, because of its simplicity and good results with not *a priori* knowledge about statistical data distribution. However, for practical problems, this rule has some drawbacks. In this work we face the solution of two of the most important: intolerance to noisy objects and high computational cost for classification.

In the related literature many methods can be found, with dissimilar behavior in different databases. A more recent approach is related with synergic methods combination, where some methods are combined, obtaining a superior behavior.

In this work two new editing methods and two new condensing methods are proposed. They have a good individual behavior and a very good combined behavior. All of them can efficiently and correctly deal with mixed and incomplete data. A significant modification of Pareto frontier, when including our methods in the combinations, shows an improvement in the state of the art.

A deeper study in different combination strategies reveals some properties of each one. The relationship between the quality of results and class mixing were also determined. The degree of class mixing was properly measured using metadata.

Palabras Clave

Selección de objetos, Combinación de métodos de selección de objetos, Clasificación supervisada, Regla de vecinos más cercanos, Datos mezclados e incompletos.

Keywords

Object selection, Object selection methods combination, Supervised classification, Nearest Neighbor rule, Mixed and incomplete data

ÍNDICE

INDICE DE FIGURAS

INDICE DE TABLAS

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 comparación 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. Por lo tanto, no todas las combinaciones son sinergias y en este trabajo se utilizará esta diferencia entre ambos conceptos.

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.

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 aeroganma 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.

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:

- 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].
- 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 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.

Dividir los datos, utilizando un clasificador apropiado para cada tipo de atributo

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).

Construir disimilaridades heterogéneas

Las disimilaridades heterogéneas trabajan directamente con datos mezclados, utilizando criterios de comparación de atributo diferentes 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 para trabajar directamente con datos mezclados de eficacia probada en problemas prácticos [24, 25].

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].

1.3 Objetivos de la tesis

El objetivo de la tesis es crear nuevos métodos de edición y condensación, cualitativamente superiores a los existentes en cuanto a su uso sinérgico, en combinaciones entre ellos y con otros métodos. Los métodos propuestos deben trabajar eficaz y eficientemente con datos mezclados e incompletos.

Otro objetivo está dirigido a estudiar la influencia del orden de aplicación de los métodos de selección en la calidad del resultado.

2 Revisión Bibliográfica

2.1 Conceptos Básicos

2.1.1 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 *m* atributos $\chi = \{\chi_1, \chi_2, \ldots \chi_m\}$, que son extraídos del problema. Cada atributo está definido en un dominio $D_i = dom(\chi_i)$. La función $\begin{matrix} \chi_i : U \to D_i \\ o \to \chi_i(o) \end{matrix}$ asocia a cada objeto el valor $\chi_i(o)$ de su atributo χ_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 su descripción, pudiéndose seleccionar el significado real del contexto.

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

2.1.2 Funciones de comparación entre objetos

Las funciones de comparación 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].

Las funciones más utilizadas para datos numéricos se resumen en la Tabla 1. [34]

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.

Tabla 1. Distancias más utilizadas. Considere x y y como dos objetos, descritos por m atributos numéricos. x_i y y_i es el valor del i-ésimo atributo de los objetos x y y respectivamente.

<table>
<tr><td>Minkowsky:
$D(x,y)=\left(\sum_{i=1}^{m}|x_i-y_i|^r\right)^{1/r}$</td><td>Euclidean:
$D(x,y)=\sqrt{\sum_{i=1}^{m}(x_i-y_i)^2}$</td></tr>
<tr><td>Manhattan / city-block:
$D(x,y)=\sum_{i=1}^{m}|x_i-y_i|$</td><td>Camberra: $D(x,y)=\sum_{i=1}^{m}\left|\frac{x_i-y_i}{x_i+y_i}\right|$</td></tr>
<tr><td>Chebychev: $D(x,y)=\max_{i=1}^{n}|x_i-y_i|$</td><td>Mahalanobis:
$D(x,y)=[\det V]^{1/m}(x-y)^T V^{-1}(x-y)$
donde V es la matriz de covarianza de los atributos.</td></tr>
<tr><td colspan="2">Cuadrática: $D(x,y)=(x-y)^T Q(x-y)=\sum_{j=1}^{m}\left(\sum_{i=1}^{m}(x_i-y_i)q_{ji}\right)(x_j-y_j)$
donde Q es una matriz de pesos definida positiva, específica del problema</td></tr>
<tr><td>Correlación:
$D(x,y)=\frac{\sum_{i=1}^{m}(x_i-\bar{x}_i)(y_i-\bar{y}_i)}{\sqrt{\sum_{i=1}^{m}(x_i-\bar{x}_i)^2\sum_{i=1}^{m}(y_i-\bar{y}_i)^2}}$</td><td>donde $\bar{x}_i$ y $\bar{y}_i$ es el valor medio de los respectivos atributos</td></tr>
<tr><td>Chi-cuadrada:
$D(x,y)=\sum_{i=1}^{m}\frac{1}{sum_i}\left(\frac{x_i}{size_x}-\frac{y_i}{size_y}\right)^2$</td><td>donde sum_i es la suma de todos los valores del atributo i que aparecen en la matriz de entrenamiento, y $size_x$ es la suma de todos los valores del vector x</td></tr>
<tr><td>Correlación de ranking de Kendall:
$D(x,y)=1-\frac{2}{n(n-1)}\sum_{i=1}^{m}\sum_{j=1}^{i-1}\text{sign}(x_i-x_j)\text{sign}(y_i-y_j)$</td><td>donde $\text{sign}(x)$ es -1, 0 o 1 según x es negativo, 0 o positivo, respectivamente</td></tr>
</table>

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

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

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 comparación de valores de atributo d_a fue substituida por la utilizada en VDM [32]:

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

2.2 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 de hacerlo: 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.

2.2.1 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 (Ver apéndice 1, para una descripción formal). 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. El riesgo de Bayes es el mínimo error teórico que puede cometer un clasificador, basado en las distribuciones estadísticas de los datos. 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 [43]. 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 [44] 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 IB-2 (Instance Based), inicialmente conocido como Growth Additive[45], 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 [46] 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 [46] se experimenta con bases de datos de hasta un millón de objetos, con dos atributos).

En 1972 Swonger [47] 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 [48] 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, 49]. 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 [50] (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. Una descripción formal en pseudo-código puede ser encontrada en el Apéndice 1.

El problema general de la obtención del subconjunto mínimo consistente ha generado muchos intentos de solución, aunque Wilfong [51] demostró que es un problema *np*-completo.

También los grafos de proximidad [52] han sido utilizados para condensar [53]. 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 [54]). Es el dual de una triangulación de Delaunay[55]. 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 [56] crearon una forma más eficiente de calcularlo, y lo aplicaron a la

selección de objetos, con resultados superiores al Quick Hull [57], el algoritmo más eficiente para calcular este grafo en espacios *n*-dimensionales.

- Grafo de vecinos relativos (utilizado en [20, 54]). 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, 54]). 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 [52].

Muchos autores han reportado métodos con un fuerte componente aleatorio. El más simple, la búsqueda al azar [58], 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, 59], dando buenos resultados, al igual que el ascenso de colina con mutaciones aleatorias [60]. Este tipo de estrategias han sido evaluadas y comparadas en varios artículos [3, 8, 61]. 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 [62]. 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 (ver Anexo 1 para la descripción formal del algoritmo). 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 comparación 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 [63]. 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 [49] desarrollaron un método basado en búsqueda tabú, un método meta-heurístico ampliamente utilizado para resolver problemas combinatorios [64]. 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 [65], 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 [66] (Compact Set Editing), un algoritmo basado en conjuntos β-compactos [67]. En este trabajo se introduce el concepto de consistencia con respecto a las subclases, que es 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 [67](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. La descripción formal del algoritmo en pseudocódigo puede encontrarse en el Apéndice 1.

También las máquinas de vectores soporte (SVM) han sido utilizadas para esta tarea [68]. 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 [69] creó un algoritmo utilizando la teoría de las relaciones estructurales grises (*Gray Relational Structures*, en inglés) [70]. 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*. [71] 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.

2.2.2 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 [72]. 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 [73] 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 [74].

En 1998, Bezdek *et al*. [75], 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 [74] 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) [76], Decision Surface Mapping (DSM) [77], Learning Vector Quantization with Training Counters (LVQ-TC) [78] y Vector Quantization (VQ) [79]. 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 [80] 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

2.3 Métodos de Edición

El ENN (Edited Nearest Neighbor) es el primer método de *edición basada en el erro*r 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. El algoritmo formal en pseudocódigo puede ser encontrado en el Apéndice 1.

En 1976 Tomek propone una variante, conocida como All-kNN [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 [81] 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.

2.4 Combinación de Métodos de Selección de Objetos

Como se mencionó en la introducción la diferencia de propósito y de características entre la edición y la condensación sugieren una relación sinérgica entre ambas. Aunque se han realizados pocos estudios en específico sobre el tema [16], muchos métodos reportados en la actualidad son combinaciones.

Aha *et al*. en 1991 [41] introdujeron el algoritmo de aprendizaje incremental IB3 (Instance Based 3). Éste, a diferencia de IB2 es capaz de eliminar objetos ruidosos, por lo que presenta mejores resultados, tanto en eficacia como en poder de condensación. IB3 almacena el número de clasificaciones correctas e incorrectas de cada objeto. Estos valores son utilizados para estimar cómo sería el comportamiento futuro de cada uno. Finalmente una prueba de significancia determina qué objetos son buenos para la clasificación y cuáles son posible ruido (para ser eliminados). Este método, al funcionar de manera incremental, es muy eficiente, pero depende fuertemente del orden de presentación de los objetos. Objetos muy importantes pueden ser descartados inicialmente, cuando el conocimiento previo de las clases está pobremente definido [15].

En 1997 Wilson [63] propuso una familia de métodos de selección de objetos, conocidos como DROP (Decremental Reduced Optimization Procedure). Entre las 5 variantes propuestas el DROP-3, es una combinación de ENN + DROP-2. Esta notación con el signo de suma (+) será utilizada en todo el documento para indicar una combinación de dos métodos, donde el que está a la izquierda es aplicado primero, y el segundo se aplica sobre el resultado del primero.

Otra combinación utilizando el DROP-2, pero sustituyendo el ENN por su propio método de edición, fue enunciada por Li *et al*. [21]. En este método de edición se utiliza el re-etiquetado y no la eliminación. Inicialmente se buscan los objetos que pertenecen a los bordes, con una idea similar al NUN (se llaman en el artículo enemigos más cercanos). Posteriormente la clase de éstos es cambiada, si es necesario, de acuerdo al voto mayoritario de sus vecinos. Finalmente, DROP-2 es aplicado para eliminar los objetos redundantes.

En 1999 Brighton y Mellish [82] desarrollan ICF (Iterative Case Filtering). Este algoritmo consiste en la repetición de una regla de eliminación simple, mientras al aplicarla se produzcan eliminaciones de objetos. En cada iteración los objetos a eliminar son marcados. Al terminar cada iteración todos los objetos marcados son eliminados, lográndose la independencia del orden. La regla puede resumirse en que un objeto es removido cuando más objetos aseguran su buena clasificación, que los que el propio objeto asegura. En una publicación de los mismos autores en el 2002 [15] señalan a

Drop3 y a ICF como los únicos métodos pertenecientes al estado del arte, aunque una afirmación tan categórica, debió haber sido respaldada por algún tipo de evidencia. De hecho, en el estudio comparativo sólo se incluyen ambos métodos, más el ENN, que es un método de edición y no de condensación. Una conclusión a la que llegan los autores es que Drop3 e ICF obtienen resultados de calidad similar.

Dasarathy *et al*. realizaron en el 2000 [16] el primer estudio, hasta donde conocemos, sobre combinación de métodos de edición y condensación. Para esto se escogen los métodos más importantes (según consideran sus autores) reportados hasta ese momento. Se elige MCS [10] y métodos de edición y condensación basados en grafos de proximidad [20, 53, 54]. Estos métodos son combinados en diferente orden, y se presentan los resultados en dos bases de datos. Las siguientes conclusiones son enunciadas:

- La edición con grafos de proximidad (PGE) alcanza el menor error, pero retiene gran parte de los objetos, por lo que no tiene un gran valor práctico.
- Los mejores resultados se obtienen con la combinación de PGE + MCS, en ese orden. En general los resultados superiores se alcanzan editando primero y después condensando.

3 Métodos Propuestos

3.1 BestByHiSCC

El primer método de condensación que proponemos en este trabajo se denomina BestByHiSCC, y utiliza la información de la estructura interna de las clases para obtener un conjunto reducido de objetos representativos. Para estructurar internamente las clases, descomponiéndolas en subclases utiliza un nuevo método de agrupamiento denominado HiSCC, que se describe en la siguiente sección.

3.1.1 HiSCC

Para la descomposición de una clase en subclases creamos un nuevo algoritmo, denominado HiSCC (Hierarchical Strongly Connected Components), perteneciente a la familia de algoritmos de agrupamiento jerárquicos aglomerativos [83]. Al igual que el resto de los algoritmos de esta familia utiliza un método de agrupamiento multicapa para producir la jerarquía. La granularidad se incrementa con la capa, siendo la capa superior la más general y los nodos hojas los más específicos. En cada nivel sucesivo, de la raíz hacia las hojas, los vértices representan subconjuntos de sus grupos padres. En cada capa, el proceso consiste en dos pasos: construcción del grafo y rutina de cobertura. La infraestructura general puede verse en la Figura 1**.**

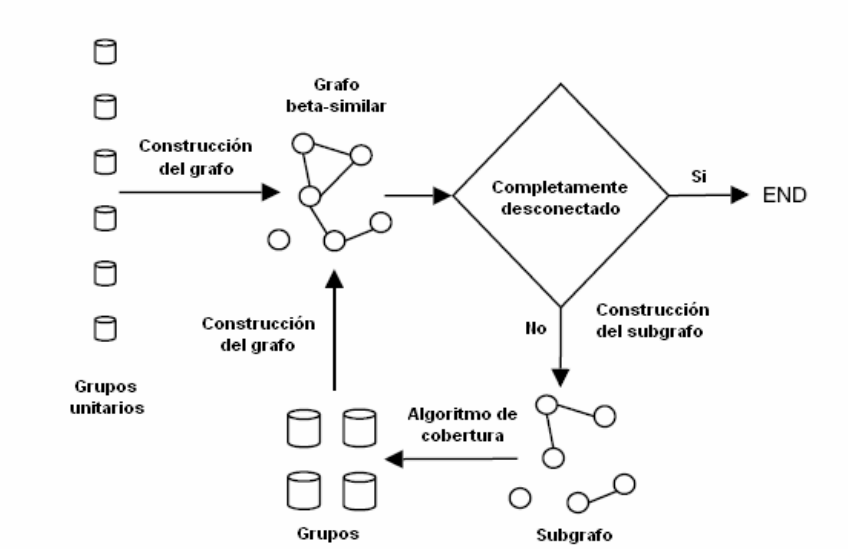


Figura 1. Infraestructura general de un algoritmo de agrupamiento jerárquico aglomerativo

Para utilizar esta infraestructura se necesita definir una disimilaridad para comparar objetos, y otra para comparar grupos de objetos. Es importante señalar que para poder utilizar esta infraestructura con datos mezclados e incompletos, se debe escoger una función apropiada (ver epígrafe 2.1.2). El algoritmo comienza con cada objeto considerado como un grupo y con ellos construye un subgrafo del grafo de β-disimilaridad inicial. Un grafo de β-disimilaridad conecta dos objetos, si el valor de la disimilaridad entre ellos es menor o igual a β. En nuestros experimentos calculamos β como la similaridad media entre todos los objetos del conjunto. Una rutina de cobertura (procedimiento que agrupa varios nodos en uno) es aplicada entonces a este subgrafo para construir los grupos de la capa inferior. A partir de los nuevos grupos, el algoritmo construye otro grafo y su correspondientes subgrafos, utilizando los grupos anteriores como nuevos vértices y evaluando su disimilaridad. Entonces se aplica nuevamente el algoritmo de cobertura para obtener los grupos de la próxima capa. El proceso se repite, hasta que se obtiene un grafo totalmente desconectado.

Nuestro algoritmo HiSCC se diferencia de los reportados anteriormente en la literatura en tres aspectos:

- Uso del máximo grupal como disimilaridad entre grupos, en vez de la media grupal. El máximo grupal es la mayor disimilaridad entre dos objetos, uno en cada grupo.
- Uso de un grafo dirigido en lugar de uno no dirigido
- Uso como rutina de cobertura las componentes fuertemente conexas del subgrafo de mínima β-disimilaridad, en vez de las componentes conexas. En una componente fuertemente conexa existe un camino dirigido entre dos cualesquiera de sus nodos.

Algoritmo 1. Algoritmo HiSCC

1. Inicialmente cada objeto constituye su propio grupo
2. $nivel = 0$
3. Construir el grafo de β-disimilaridad, G_{nivel}
4. Mientras G_{nivel} no sea totalmente desconectado
   a. Construir el subgrafo dirigido de mínima β-disimilaridad
   b. Buscar sus componentes fuertemente conexas
   c. Construir en nuevo grafo de β-disimilaridad, $G_{nivel+1}$, donde cada vértice representa una componente fuertemente conexa, y la disimilaridad entre grupos, es la máxima entre dos elementos cualesquiera, uno de cada grupo
   d. $nivel \leftarrow nivel + 1$

Estas diferencias hacen que se produzcan clusters más homogéneos, lo que posibilita elegir un representante por agrupamiento muy parecido al resto de los miembros del agrupamiento. Sin embargo, el algoritmo mantiene algunas propiedades deseadas, como la independencia del orden de presentación de los objetos, no tener que especificar de antemano la cantidad de grupos, y tener un solo parámetro para optimizar. En nuestros experimentos calculamos β como la media de todas las disimilaridades del conjunto de entrenamiento original.

El nuevo algoritmo BestByHiSCC consiste en la selección del objeto más representativo por cada subclase. Las subclases son encontradas mediante la estructuración de cada clase con el HiSCC. El objeto más representativo es encontrado como el de menor suma de disimilaridades con respecto a los demás de su subclase. En caso de no ser único, se escoge al más disimilar con respecto a los objetos de las subclases más cercanas.

Este algoritmo no tiene en cuenta la interacción de cada clase con el resto, por lo que si éstas están muy entremezcladas, puede dar resultados no representativos del conjunto original. No obstante al ser combinado con métodos de edición, logra niveles de reducción muy superiores al resto de los métodos, con poca afectación de la eficacia del clasificador resultante.

Algoritmo 2. Algoritmo BestByHiSCC

1. Descomponer cada clase en subclases utilizando HiSCC
2. Elegir el representante de cada subclase, por los siguientes criterios:
   a) La suma de las disimilaridades entre él y el resto de los miembros de su subclase es mínima, es decir, el representante es el objeto al que más se parecen el resto de los integrantes de la subclase
   b) En caso de no ser único, seleccionar el que la suma con los miembros de las subclases más cercanas sea máxima
3. El resultado final es la unión de todos los representantes de todas las subclases de cada clase

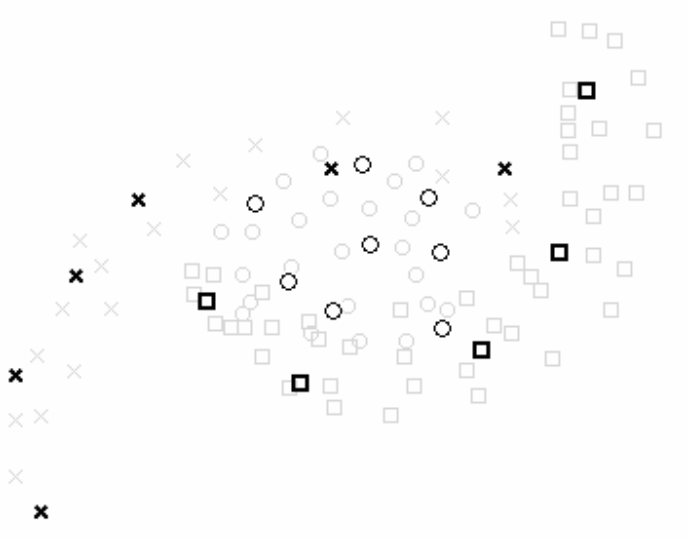


Figura 2. BestByHiSCC en base de datos Bananas&Círculo

En este capítulo se utilizan algunas bases de datos sintéticas con objetos descritos por dos rasgos numéricos enteros. Éstas fueron generadas con el propósito de mostrar visualmente algunas características de los resultados arrojados por los métodos, o para comparar los resultados de varios de ellos, y por ello no se utilizan en experimentaciones.

En la Figura 2 puede apreciarse la aplicación de BestByHiSCC a una base de datos sintética. Esta base de datos está formada por tres clases, dos de ellas distribuidas en forma de banana (cruces y cuadrados), y la tercera distribuida en forma de círculo, en el

centro de ambas bananas (círculos). Se aprecia que son seleccionados pocos objetos, pero que representan eficazmente los puntos originales.

3.2 CSESupport

El segundo algoritmo de condensación propuesto, está diseñado para la búsqueda de una aproximación mejor al subconjunto mínimo consistente, que los previamente reportados en la literatura. Utiliza un algoritmo inspirado en el del CSE original [66], utilizando un grafo distinto, un nuevo criterio de evaluación de objetos y realizando varias iteraciones.

Se basa fundamentalmente en la construcción de un grafo de soporte, de ahí su nombre. Un grafo de soporte es un grafo dirigido, donde cada arco conecta a un objeto con todos aquellos de su propia clase más similares que su NUN [50]. El NUN de un objeto es su vecino más similar de clase diferente. Por tanto un arco del objeto x al objeto y significa que y es más similar a x que el NUN de x, por lo que de quedar y en el resultado, garantiza la buena clasificación de x, y por tanto lo *soporta*. De esta manera los vértices con mayor grado de entrada son los que soportan a la mayor cantidad de objetos, y por tanto, los más importantes.

El CSESupport elimina a los objetos menos importantes, garantizando la consistencia del resultado mediante una estrategia de marcado. Esta estrategia consiste en que cuando un objeto se va a eliminar, *marca* a todos los que lo soportan, teniendo que quedar alguno de éstos en el resultado final. De esta forma, cuando un objeto es el último con una marca, tiene que ser incluido en el resultado. De manera análoga, si algún objeto no tiene ningún sucesor, no tiene ningún objeto que lo soporte y por tanto tiene que pertenecer al resultado.

Este método parte de un conjunto consistente (en la primera iteración, todo el conjunto inicial) y construye un subconjunto consistente. Esto lo realiza mediante la eliminación de los objetos menos importantes (según la heurística utilizada), manteniendo a los objetos que los soportan. Si en este proceso se eliminó algún objeto entonces es posible que los NUN de algunos objetos hayan cambiado, por lo que se vuelve a iterar para buscar una mayor reducción. Nótese que el grafo de soporte (el que garantiza la

consistencia) se construye con todos los objetos de la matriz original, para garantizar la consistencia con respecto a los objetos originales en cada una de las iteraciones.

Algoritmo 3. CSESupport
1. $E \leftarrow T$
2. Construir el grafo dirigido de soporte G, donde los vértices son los elementos de T, y cada vértice se conecta con aquellos que estén más cercanos (según la disimilaridad utilizada) que su NUN [50] en E.
3. $E_{new} \leftarrow \phi$, nuevo conjunto consistente a construir
4. Eliminar los arcos entre objetos de diferentes clases, moviendo a E_{new} los objetos que aparecen en el origen del arco. Para esto se utiliza el procedimiento $move(x)$.
5. Mover a E_{new} mediante $move(x)$ los vértices que quedaron sin ningún antecesor en G.
6. Mover a E_{new} mediante $move(x)$ cualquier vértice que sea el último marcado para no eliminación simultánea.
7. Eliminar el vértice menos importante en el grafo, mediante $discard(x)$. Este se determina por ser el que menos antecesores tiene en G, y en caso de no ser único, el que, de éstos, menos objetos lo hayan marcado para no eliminación simultánea.
8. Si aún quedan vértices en G, ir a 5
9. Si $E_{new} \neq E$, hacer $E \leftarrow E_{new}$ e ir a 2
10. Terminar, el resultado es E

Procedimiento $move(x)$
m1. Los antecesores de x en el grafo son marcados, porque ya están asegurados al estar x en el resultado. Éstos, al eliminarse, no marcarán a sus sucesores para no eliminación simultánea
m2. Todas las marcas de no eliminación simultánea de objetos que la compartan con x son eliminadas
m3. Eliminar a x de G

Procedimiento $discard(x)$
d1. Si x no está marcado como previamente asegurado, marcar todos los sucesores de x en G con una marca de no eliminación simultánea
d2. Eliminar a x de todas las marcas de no eliminación simultánea en las que aparezca
d3. Eliminar a x de G

De la Figura 3 a la Figura 6, puede verse la comparación de resultados entre el MCS [10], el CNN [7], el RNN [9] y nuestros CSESupport en una base de datos artificial de ejemplo. Estos métodos se seleccionaron pues son los más referenciados en la literatura para la búsqueda del subconjunto mínimo consistente. La base de datos está formada por objetos de tres clases, con una zona de gran mezcla entre las clases. Puede verse cómo el

CSESupport logra encontrar un subconjunto consistente de menor tamaño que el MCS, con un costo computacional similar. En el caso del RNN, permutando aleatoriamente los objetos, y después de varias ejecuciones es posible hallar un subconjunto consistente menor. Por el alto costo computacional del RNN, este proceso en bases de datos de mayor tamaño puede ser demasiado costoso.

El CSESupport encuentra frecuentemente subconjuntos consistentes con menor cantidad de objetos que el MCS. En nuestra opinión esto se debe a que sigue una estrategia de eliminar el objeto menos importante, al contrario del MCS que utiliza la estrategia de seleccionar al más importante. Esto coincide con varios resultados obtenidos en otros dominios, en que los métodos de adición incremental obtienen una calidad del resultado inferior a los de eliminación decremental.

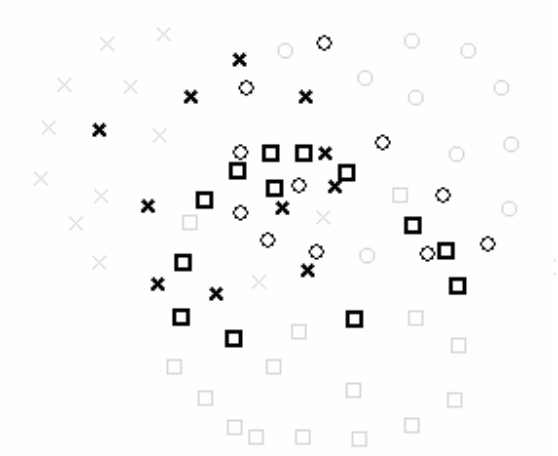


Figura 3. MCS en base de datos Venn (35)

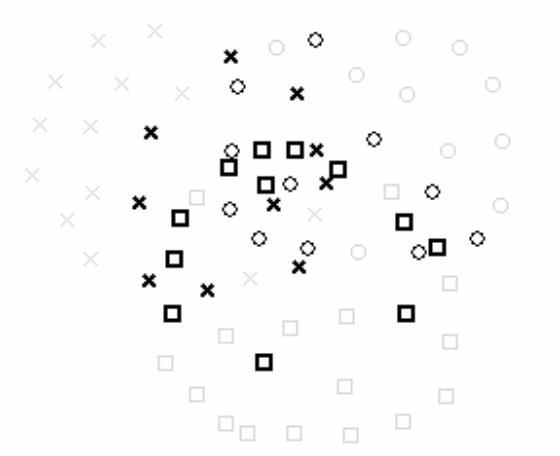


Figura 4. CSESupport en base de datos Venn (33)

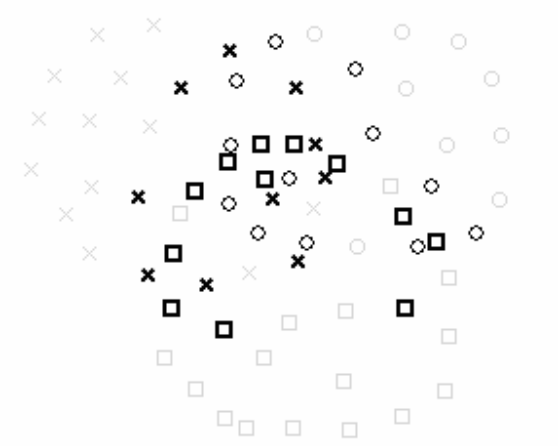


Figura 5. RNN en base de datos Venn (34)

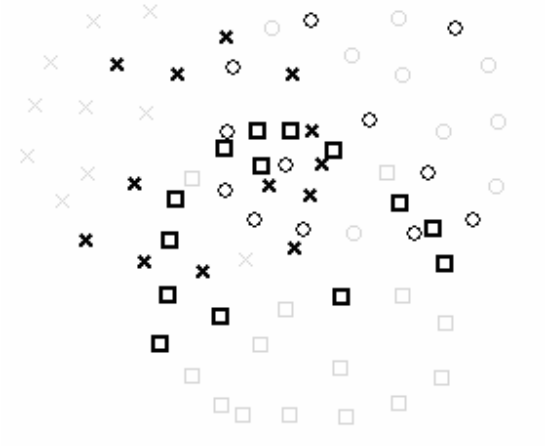


Figura 6. CNN en base de datos Venn (39)

3.3 CSEdit

El análisis de los métodos de edición reportados en la literatura mostró algunas deficiencias importantes:

- Dependen críticamente de algún parámetro, como el caso del valor de k en el ENN y el All-kNN
- En la selección de los vecinos de un objeto no se toma en cuenta el valor de la disimilaridad, por lo que un objeto puede ser considerado vecino de objetos muy disimilares. Este es el caso, por ejemplo, de los métodos basados en grafos de proximidad
- En las zonas donde el error de Bayes es muy elevado, no realizan una eliminación efectiva de objetos de clase dudosa (ver Figura 7 y Figura 8)
- En problemas con pocos objetos, pueden eliminar clases completas. Un ejemplo de esto es el Multiedit

Para tratar de mitigar estas deficiencias creamos dos nuevos métodos de edición, que utilizan la información de un grafo G de máxima similaridad, para detectar los objetos candidatos a ser eliminados. Un grafo de máxima similaridad es un grafo dirigido, donde un arco del objeto x al objeto y significa que y es el objeto más similar a x. Este tipo de grafos no necesita de ningún otro parámetro para su construcción (que no sea la similaridad) y los objetos sólo se conectan con aquellos con los que son más similares. En las zonas donde el error de Bayes es alto, los vecinos de cada objeto proveen información útil para estimar el grado de certeza que tenemos sobre su clasificación correcta.

El primer método, denominado CSEditA, elimina los objetos que tengan al menos un sucesor en el grafo de máxima similaridad G de clase diferente. Estos objetos tienen otro objeto, al que ellos son los más parecidos, de clase diferente, lo que presupone cierto nivel de duda sobre lo correcto de su clase actual.

El segundo método, nombrado CSEditB, elimina los objetos mal clasificados por un voto mayoritario de todos sus vecinos en el grafo de máxima similaridad G, o sea, sus

antecesores y sucesores. Nótese que aunque la información que brinda tener sucesor y un antecesor de clase diferente en el grafo es semánticamente diferente, es igualmente válida para decidir de qué objetos se tiene suficiente evidencia sobre lo correcto de su clase. De esta forma los objetos que están rodeados de otros de clases diferentes son siempre eliminados, mientras que los objetos de las fronteras entre las clases son eliminados según el grado de certeza de su pertenencia a la clase que tiene asignada.

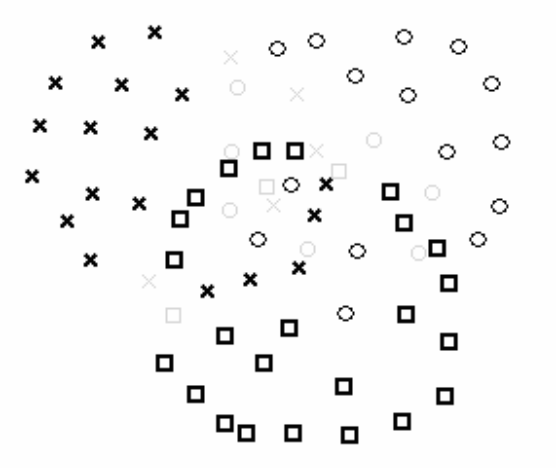


Figura 7. ENN en base de datos Venn

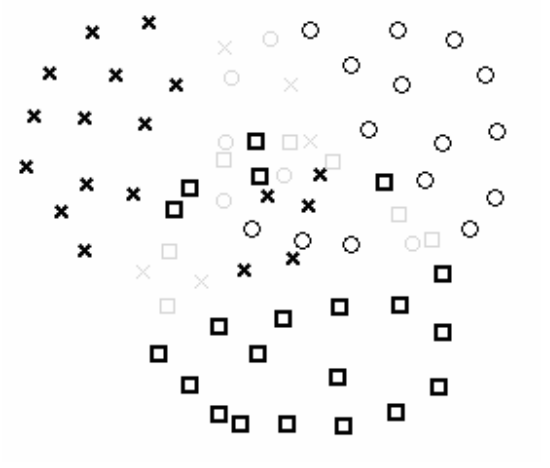


Figura 8. RNE en base de datos Venn

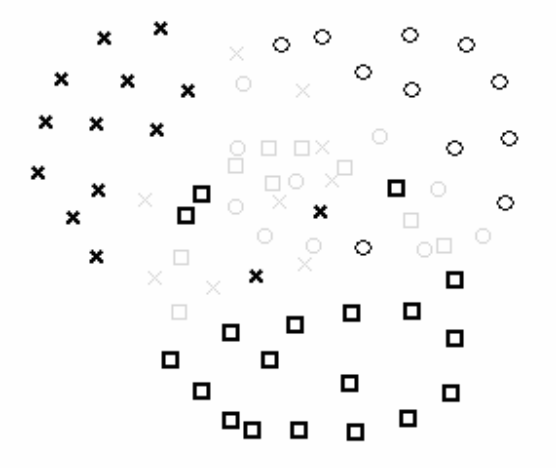


Figura 9. CSEditA en base de datos Venn

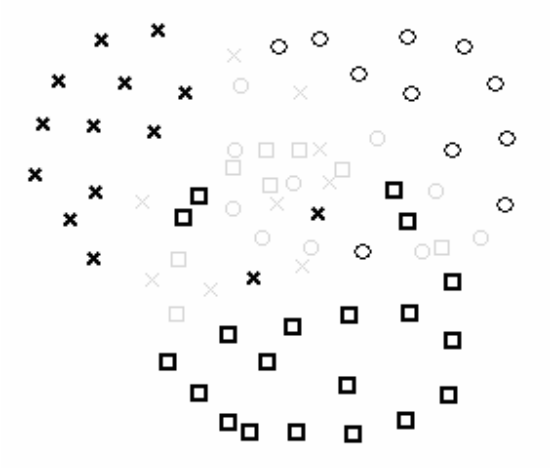


Figura 10. CSEditB en base de datos Venn

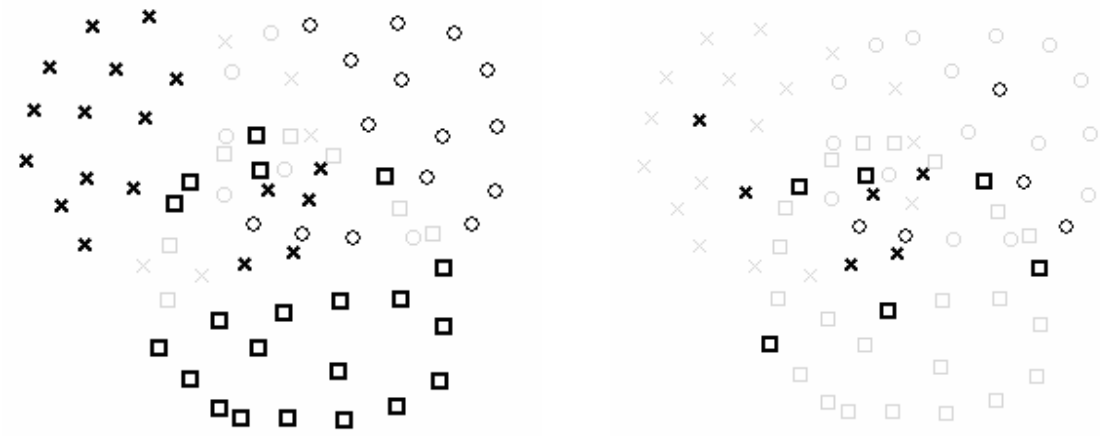
De la Figura 7 a la Figura 10 se puede ver la comparación de la aplicación del ENN [11], el RNE [20] y nuestros dos métodos. Cómo se puede ver nuestros métodos eliminan casi

todos los objetos de la zona de mayor riesgo, quedándose con los objetos de clasificación más segura, mientras que tanto el ENN como el RNE no pueden eliminar muchos de estos objetos.

3.4 Combinación de Métodos

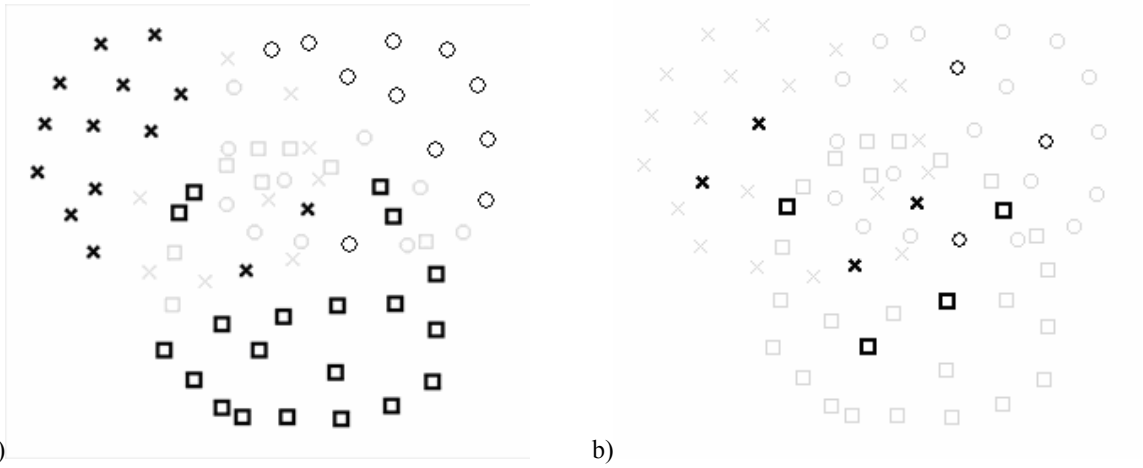
Como se mencionó en la introducción, las combinaciones de métodos de selección de objetos no han sido muy estudiadas. En esta sección presentamos algunos resultados de combinaciones de métodos, tanto de los propuestos en este trabajo como en trabajos previos, para mostrar visualmente el efecto sinérgico de algunas combinaciones.

El resultado de la selección de objetos con algunas combinaciones de métodos se incluye de la Figura 11 a la Figura 14. Se presentan, por separado, el resultado de la aplicación del primer método, y la del segundo, aplicado sobre el conjunto resultante del primero. En la Figura 11 aparece la combinación RNE + MCS, la de mejores resultados en el estudio de Dasarathy [16]. De la Figura 12 a la Figura 14 se muestran algunas combinaciones que incluyen a nuestros métodos, con buenos resultados en los experimentos.



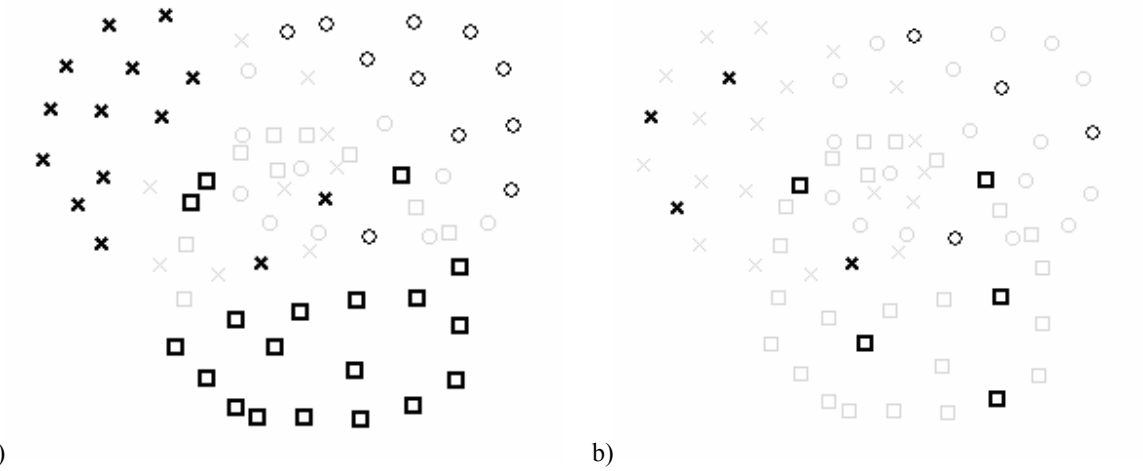
a) b)

Figura 11. a) RNE y b) RNE + MCS (17 objetos)



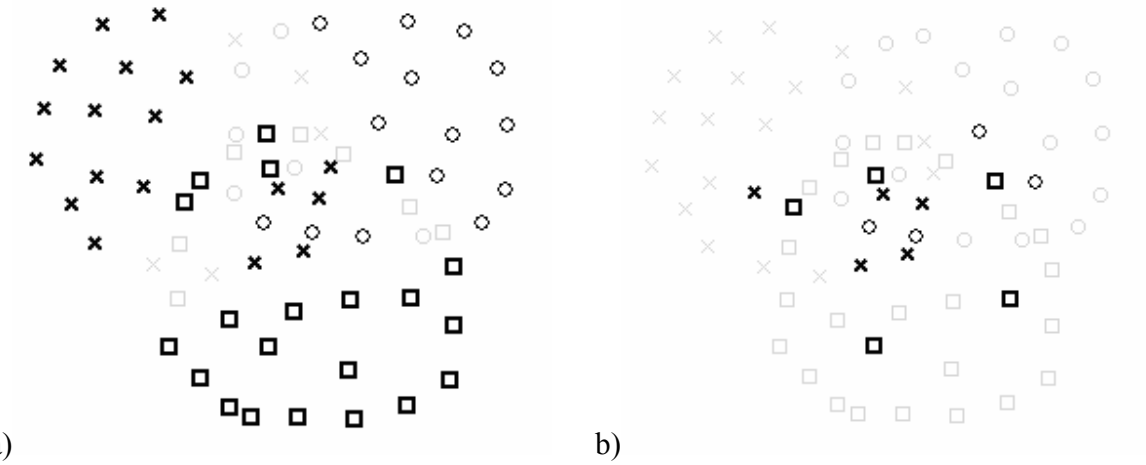
a) b)

Figura 12. a) CSEditB y b) CSEditB + CSESupport (11 objetos)



a) b)

Figura 13. a) CSEditA y b) CSEditA + BestByHiSCC (13 objetos)



a) b)

Figura 14. a) RNE y b) RNE + CSESupport (14 objetos)

Como puede apreciarse las combinaciones RNE + MCS (la mejor encontrada por Dasarathy [16] y RNE + CSESupport obtienen resultados con muchos objetos, y estos están ubicados en la región central, donde el entremezclado de las clases es mayor. Esto

se debe, parcialmente, a que el RNE no es capaz de eliminar correctamente estos objetos, mientras que tanto el CSEditA como el CSEditB sí. Por otra parte las combinaciones CSEditB + CSESupport y CSEditA + BestByHiSCC generan resultados con menos objetos, pero más representativos del conjunto original.

4 Experimentación y Resultados

4.1 Bases de Datos

Para los experimentos se seleccionaron 10 bases de datos del repositorio de la Universidad de California en Irvine (UCI) [23]. Éstas se escogieron con características diferentes, para poder apreciar el desempeño de los métodos ante diferentes distribuciones y tipos de los datos.

Un resumen de las características de cada base de datos aparece a continuación. En la Tabla 2 aparece el nombre de la base de datos, tal y como se utilizará en el resto del documento. A continuación el campo **Validación cruzada** (Valid. Cruz) que indica si la base de datos ya venía dividida en entrenamiento y control en el repositorio, o si se utilizó validación cruzada. Posteriormente el tamaño de la base de datos, dividido en **Entrenamiento** y **Prueba**, o sólo el total, según el valor del campo anterior. Finalmente aparece la cantidad de atributos numéricos y no numéricos.

Tabla 2. Algunas características de las bases de datos utilizadas en las experimentaciones numéricas

Base de datos	Valid. Cruz.	Tamaño de BD			Cantidad de Atributos	
		Ent.	Prueba	Total	Numéricos	No Numéric.
Annealing	-	798	100	-	6	32
BreastCancerWisconsin	si	-	-	369	9	-
cmc	si	-	-	1473	9	-
credit-screening	si	-	-	690	6	9
cylinder-bands	si	-	-	512	20	20
Iris	si	-	-	150	4	-
Phoneme	si	-	-	5404	5	-
thyroidDiseaseAnn	-	3772	3428	-	6	15
wdbc	si	-	-	569	30	-
wpbc	si	-	-	198	32	-

En la Tabla 3 aparecen otros datos de interés. La segunda columna expresa si la base de datos presenta ausencia de información en algunos de sus objetos. Al final aparecen los porcentajes respectivos de objetos por clase.

Tabla 3. Otras características de las bases de datos utilizadas en las experimentaciones numéricas

Base de datos	Ausencia de Información	Porcentaje de objetos por clase
Annealing	si	**76% / 4% / 1% / 8% / 11%**
BreastCancerWisconsin	si	**65.5% / 34.5%**
cmc	-	**42.7% / 22.6% / 34.7%**
credit-screening	si	**44.5% / 55.5%**
cylinder-bands	si	**60.9% / 39.1%**
Iris	-	**33% / 33% / 33%**
Phoneme	-	**71% / 29%**
thyroidDiseaseAnn	-	**92.5% / 5.1% / 2.5%**
wdbc	-	**62.7% / 37.3%**
wpbc	-	**76.3% / 23.7%**

4.2 Métodos de Comparación de Resultados

En esta sección se exponen los métodos utilizados para la comparación de los resultados experimentales. En cada base de datos se utilizó validación cruzada con 10 hojas (10 fold cross validation), excepto en aquellas que ya están divididas en entrenamiento y control en el repositorio. La validación cruzada permite una estimación adecuada del error de un método, utilizando para el control objetos que el clasificador no ha utilizado en su proceso de entrenamiento.

Dos magnitudes fundamentales fueron estimadas en cada experimento: la taza de retención y el error. La tasa de retención es la fracción de objetos resultantes de la aplicación del método de selección, con respecto a la cantidad de objetos originales. El error se estima con la cantidad de objetos mal clasificados por un 1-NN, con respecto al total. La tasa de retención da una medida directa del ahorro de tiempo de clasificación del clasificador resultante, mientras que el error permite medir en qué medida este ahorro de tiempo compromete la calidad del resultado.

4.2.1 Distancia Euclidiana del Resultado

Una forma de balancear los objetivos de máxima reducción con mínimo error, frecuentemente opuestos, consiste en la utilización de la distancia euclidiana entre cada par (Retención, Error) y el punto (0,0). A medida que un resultado está más cercano del punto (0,0), logra una mayor reducción del número de objetos, con un menor error [16]. En el caso de una aplicación práctica, si uno de los objetivos es más importante que el otro, se puede sustituir por una distancia pesada.

4.2.2 Frontera de Pareto

Para evaluar la calidad del resultado de un problema multi-objetivo, tradicionalmente se ha utilizado la frontera de Pareto [84]. La misma está compuesta por todos los resultados que son Pareto – óptimos. Un resultado Pareto - óptimo en un conjunto es aquél que no se puede encontrar ningún otro en el mismo conjunto que lo supere o iguale en todos los parámetros a la vez.

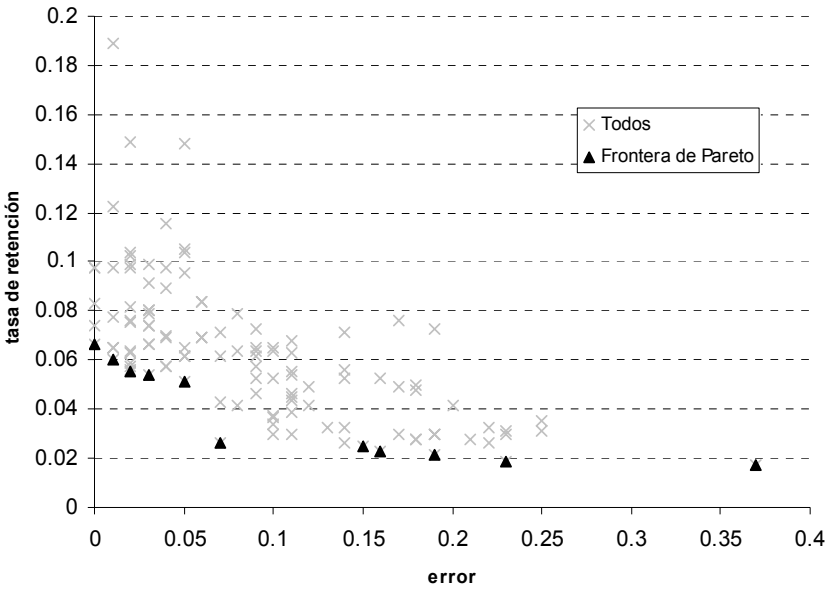


Figura 15. Ejemplo de resultado de métodos de selección, destacando la frontera de Pareto

En el problema que abordamos en este trabajo, los dos objetivos a minimizar son el porcentaje de retención y el error. En este trabajo un resultado es Pareto – óptimo, si no existe ningún otro resultado que tenga, simultáneamente, una retención y un error menor (Figura 15).

No es difícil demostrar que el objeto con la menor distancia euclidiana al punto (0,0) es siempre Pareto–optimal.

4.2.3 Validación cruzada

La validación cruzada, también llamada estimación de rotación, es la práctica estadística de particionar una muestra de datos en subconjuntos, de tal forma que el análisis es

inicialmente realizado en un subconjunto, mientras el(los) otro(s) son utilizados para confirmar y validar el análisis inicial [6]. Existen diferentes variantes de validación cruzada, pero en este trabajo utilizaremos la más utilizada en los artículos dedicados a la selección de objetos, la validación cruzada en 10 partes. Ésta consiste en particionar la muestra original en 10 subconjuntos, y utilizar uno como control y los 9 restantes como entrenamiento. Este proceso se realiza con todos los subconjuntos y el resultado es promediado para obtener una estimación del error total.

4.3 Métodos de selección de objetos utilizados

Para los experimentos se seleccionaron los métodos más utilizados en las comparaciones reportadas en la literatura. Se incluyen métodos tradicionales, así como del estado del arte. Estos métodos de selección de objetos fueron descritos en detalles en la revisión bibliográfica, y sus descripciones formales se encuentran en el Anexo 1.

Los métodos de condensación son el CNN [7], MCS [10], condensación con grafos de proximidad (PGC) [53], TIBL [62], DROP-3 [22], CSE [66], BestByHiSCC y CSESupport. Los métodos de edición utilizados son el ENN [11], la edición con grafos de proximidad (PGE) [53], CSEditA y CSEditB.

Todos los métodos fueron ejecutados en cada base de datos, así como todas las combinaciones secuenciales de dos métodos, incluyendo cada método consigo mismo.

4.4 Alteración de la frontera de Pareto

Como ya analizamos en el Epígrafe 4.2.2, el cálculo de la frontera de Pareto es una medida adecuada para evaluar la calidad de resultados, en problemas multiobjetivos. Un acercamiento de la frontera de Pareto al punto (0,0) puede ser interpretado como un aporte significativo al estado del arte en la materia. Por otra parte, la inclusión de un resultado en la frontera de Pareto, implica que está a nivel de los resultados de los métodos que forman el estado del arte.

En este epígrafe mostraremos los cambios ocasionados a la frontera de Pareto al incluir nuestros métodos, tanto con los métodos originales, como con sus combinaciones.

En las figuras de este epígrafe, y en los anexos, se representan las fronteras de Pareto, considerando dos magnitudes. En el eje horizontal se encuentra el error de clasificación,

evaluado con un conjunto de prueba que no participa en el entrenamiento. En el eje vertical aparece la tasa de retención, calculada como la razón de la cantidad de objetos seleccionados entre el total. Con un (◊) se representa la frontera original, es decir, la frontera calculada sin incluir ninguno de los métodos propuestos en este trabajo. Por otra parte, con una (x) se representa la nueva frontera, al incluir los métodos propuestos y sus combinaciones. En el caso que un resultado esté incluido en ambas fronteras, se representa con un símbolo combinado de los anteriores. O sea, una (x) indica un resultado que incluye al menos a uno de nuestros métodos, un (◊) un resultado que no los incluye y un símbolo combinado un resultado de la frontera original que no pudo ser superado por ninguno de la nueva frontera.

Los nuevos métodos, sin combinar, se colocan frecuentemente en la frontera de Pareto (Figura 16), logrando en varias bases de datos modificarla sustancialmente (Figura 17).

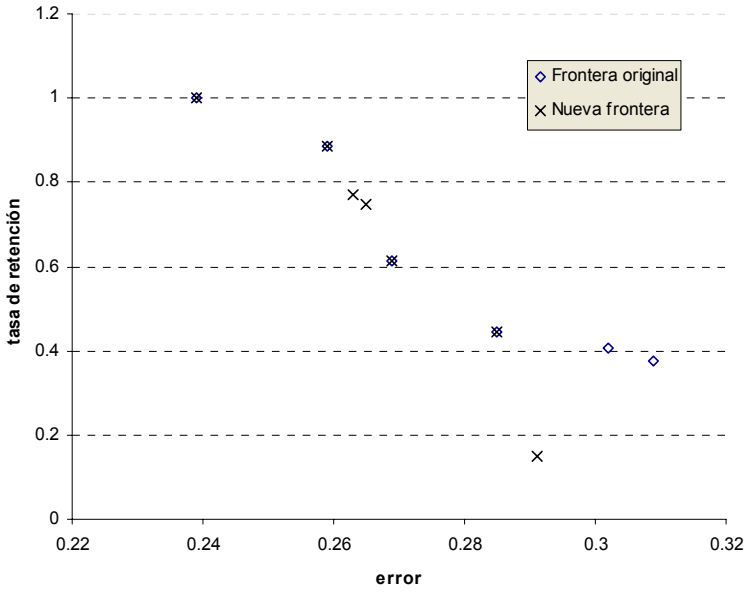


Figura 16. Cambio de la frontera de Pareto en base de datos WDBC (HVDM), con la inclusión de los métodos propuestos en este trabajo

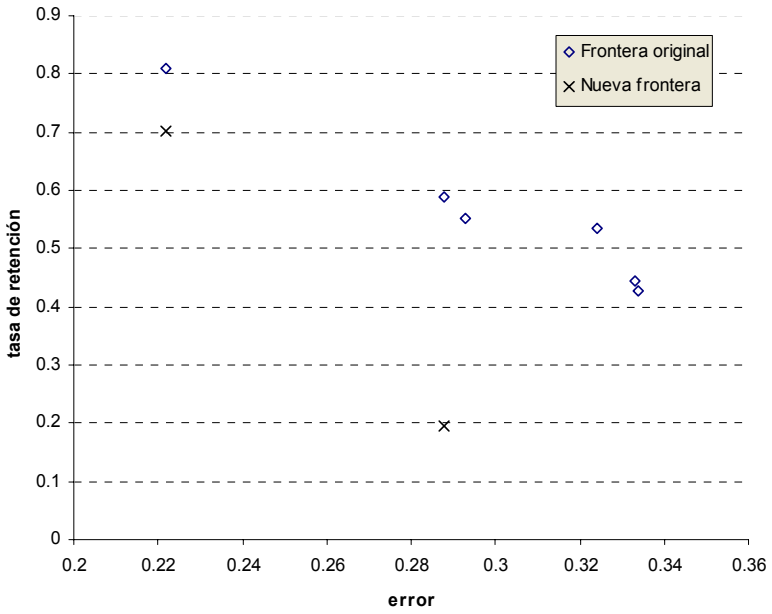


Figura 17. Cambio de la frontera de Pareto en base de datos WPBC (HVDM), con la inclusión de los métodos propuestos en este trabajo

A diferencia de los resultados obtenidos por los métodos de selección de objetos individuales, al realizar las combinaciones de métodos, en la mayor parte de las bases de datos hubo un cambio total en la frontera de Pareto, ya que ningún resultados de la frontera previa se encuentra en la nueva frontera (por ejemplo, ver Figura 18). Esto se debe a que las combinaciones con los nuevos métodos de selección de objetos creados superaron a todas las combinaciones que no los incluían. Esta modificación total de la frontera significa que para cualquier combinación de métodos sin incluir los nuevos, existe una combinación con los nuevos que la supera, con una retención y un error menor. También es interesante destacar que la enorme mayoría de los resultados en la frontera, son de combinaciones de métodos. Esto induce a pensar que no obstante lo bueno que puede ser un método individual, es posible mejorarlo mediante una combinación apropiada.

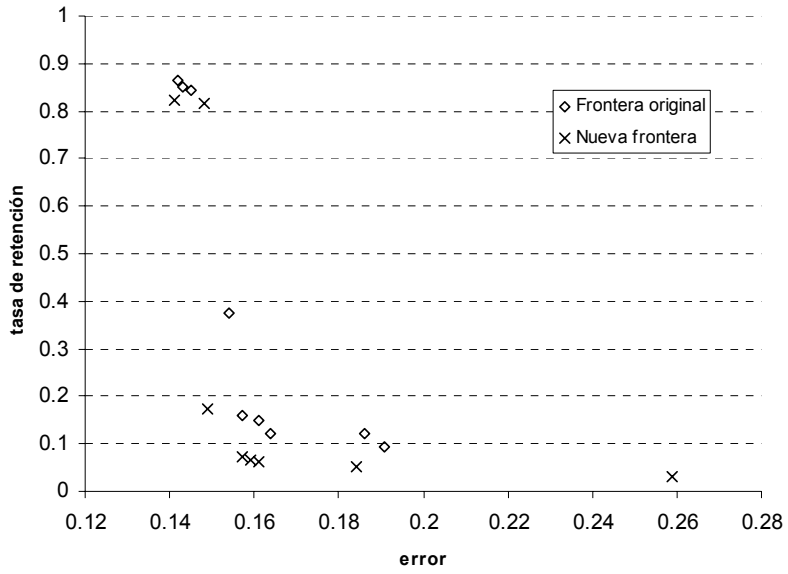


Figura 18. Fronteras de Pareto en base de datos CreditScreening (HVDM), antes y después de incluir los nuevos métodos

También en algunas bases de datos se modificó sólo una parte de la frontera, pero usualmente de manera más apreciable que con los métodos individuales (por ejemplo ver la Figura 19).

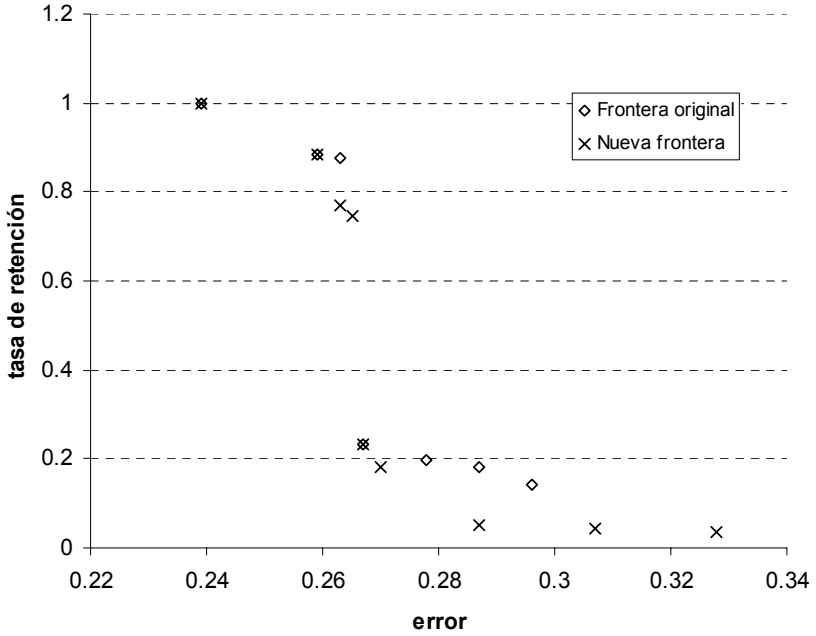


Figura 19. Fronteras de Pareto en base de datos CylinderBand (HVDM), antes y después de incluir los nuevos métodos

Es importante resaltar que, aunque en este epígrafe sólo presentamos resultados con HVDM, las mejoras ocurrieron independientemente de que la función de comparación utilizada fuera la euclidiana o HVDM, como puede apreciarse en el Anexo 2. De todos estos resultados se puede concluir que los nuevos métodos están al nivel de los del estado del arte utilizados de manera individual, y mejoran el estado del arte cuando se utilizan de manera combinada.

La modificación de la frontera de Pareto en todas las bases de datos y las combinaciones de métodos puede consultarse en el Anexo 2.

4.5 Resultados con la distancia euclidiana del resultado

Todos los métodos y combinaciones fueron ordenados según la distancia euclidiana del par (error, tasa de retención) y el punto (0,0) [16], procesándose de varias maneras, con los resultados mostrados a continuación.

Para construir la Tabla 4 se contó cuantas veces cada método aparece entre los 5 mejores resultados de cada base de datos (*Cantidad*), presentándose sólo aquellos que aparecieron más de una vez. Los valores de *Categoría* están acorde a los que aparecen en la Tabla 5. La tasa de calidad se calcula dividiendo la cantidad de apariciones, con respecto a la cantidad de veces que su categoría fue la mejor en las bases de datos (Tabla 6 y Tabla 7). Esto da una medida de calidad relativa, tomando en cuenta que no todas las estrategias son efectivas en cada base de datos.

Algunas conclusiones interesantes pueden extraerse de estos resultados. La primera de ellas es que las combinaciones con los métodos propuestos son superiores a las existentes en el estado del arte, ya que sólo una combinación (ENN + MCS) logra incluirse entre las mejores 5, en una sola base de datos (por ser una, no aparece en la Tabla 4). Otro resultado interesante, es que el 30% de estos resultados, incluyen combinaciones sólo de nuestros métodos. En resumen, del total de métodos seleccionados (15 Bases de datos * 5 mejores = 75), el 1.3% no incluye ninguno de nuestros métodos, el 68.7% incluye a uno de nuestros métodos y el 30% está formado exclusivamente por nuestros métodos.

Tabla 4. Resumen de los resultados de la distancia euclidiana entre los resultados con ambas funciones de comparación

Métodos	*Cantidad*	*Categoría*	*Tasa de Calidad*
CSEditB+CSESupport	6	EC	0.86
CSEditA+CSESupport	6	EC	0.86
CSEditB+Drop3	6	EC	0.86
MCS+CSEditB	6	CE	1.00
CSEditA+MCS	4	EC	0.56
CSESupport+CSEditB	4	CE	0.66
CSEditA+Drop3	3	EC	0.43
CSEditB+MCS	3	EC	0.43
Drop3+CSEditA	3	CE	0.5
Drop3+CSEditB	3	CE	0.5
BestByHiSCC+Drop3	2	CC	1
BestByHiSCC+TIBL	2	CC	1
CSE+CSEditB	2	CE	0.33
MCS+CSEditA	2	CE	0.33
RNE+CSESupport	2	EC	0.29

4.6 Comparación del orden de aplicación de los métodos

En esta sección brindaremos los resultados obtenidos en cuanto a la calidad de los resultados, dependiendo del orden de aplicación de los métodos. Los métodos y las combinaciones fueron etiquetados apropiadamente, según aparece en la Tabla 5.

Tabla 5. Abreviaturas de las categorías en que se clasificó cada método y sus combinaciones

Categoría	*Orden de aplicación*	*Comentarios*
E	Edición	
C	Condensación	
EC	Edición + Condensación	Primero el método de edición, y sobre su resultado, el método de condensación
CE	Condensación + Edición	Primero el método de condensación, y sobre su resultado, el método de edición
CC	Condensación + Condensación	Primero el método de condensación, y sobre su resultado, el otro método de condensación
EE	Edición + Edición	Primero el método de edición, y sobre su resultado, el otro método de edición

Para cada base de datos se obtuvieron 157 resultados (12 métodos individuales, más 12 * 12 combinaciones, más el resultado sin seleccionar objetos), clasificados en las 6 categorías anteriores. Para determinar con qué categoría se obtienen los mejores resultados en cada base de datos, los resultados fueron ordenados ascendentemente de acuerdo a la distancia euclidiana del resultado. Posteriormente se asignó a cada resultado

un premio igual a $1/Posicion$. Finalmente para cada categoría sus valores de premio fueron sumados, correspondiendo el mayor valor al mejor resultado. Este método de cálculo permite que las categorías con mayor cantidad de elementos entre los primeros (los de mejores resultados), obtengan un puntaje mayor.

Un resumen de los resultados obtenidos en las bases de datos del repositorio con la función de comparación HVDM aparece en la Tabla 6. Un resumen similar, pero con las bases de datos numéricas y la distancia euclidiana, aparece en la Tabla 7. Los valores corresponden a la suma de los premios en cada categoría. Las últimas 5 filas de la Tabla 7, así como la última columna de ambas tablas serán explicadas posteriormente.

Tabla 6. Comparación del orden de aplicación de métodos con HVDM. En negrita el mejor resultado

Base de datos	*CE*	*EC*	*CC*	*EE*	*E*	*C*	*MetaIdx*
Annealing	0.506	**3.065**	1.730	0.111	0.026	0.193	0.920
BreastCancerWisconsin	1.092	**3.275**	1.029	0.110	0.027	0.098	0.950
CMC	**2.986**	1.619	0.714	0.198	0.041	0.073	0.510
CreditScreening	**2.755**	1.897	0.758	0.110	0.027	0.084	0.830
CylinderBand	1.047	0.962	**3.328**	0.110	0.027	0.157	0.730
Iris	0.363	**3.502**	1.516	0.109	0.027	0.113	0.930
Phoneme	0.876	**3.713**	0.805	0.110	0.027	0.099	0.860
ThyroidDiseaseAnn	**2.567**	2.025	0.793	0.112	0.027	0.107	0.940
WDBC	1.016	1.143	**3.254**	0.110	0.027	0.081	0.950
WPBC	**3.623**	1.012	0.779	0.110	0.027	0.080	0.710

Tabla 7. Comparación del orden de aplicación de métodos con la distancia euclidiana. En negrita el mejor resultado

Base de datos	*CE*	*EC*	*CC*	*EE*	*E*	*C*	*MetaIdx*
BreastCancerWisconsin	1.508	**3.094**	0.803	0.110	0.027	0.088	0.960
CMC	**2.359**	2.253	0.651	0.248	0.048	0.070	0.540
Iris	0.307	**3.631**	1.453	0.110	0.027	0.102	0.950
WDBC	0.778	**3.760**	0.858	0.110	0.027	0.097	0.920
WPBC	**3.400**	1.284	0.734	0.110	0.027	0.075	0.710
GausMean0&1	**3.580**	1.102	0.720	0.128	0.028	0.073	0.631
GausMean0&2	**3.398**	1.314	0.704	0.110	0.027	0.078	0.784
GausMean0&3	**2.540**	2.111	0.753	0.110	0.027	0.090	0.909
GausMean0&4	1.453	**3.113**	0.826	0.110	0.027	0.101	0.968
GausMean0&5	0.487	**3.841**	1.042	0.110	0.027	0.122	0.990

Los resultados no fueron promediados, pues nos interesa estimar cuál estrategia genera resultados que se ubiquen en los primeros lugares, no las que generen mejores resultados medios. De promediarse es posible que una estrategia con resultados siempre medios, dé valores mayores que una que tiene los primeros y los últimos lugares.

El mayor inconveniente de la suma consiste en que no hay la misma cantidad de métodos en cada categoría. Para mitigar este efecto se escogió la asignación de pesos por el inverso de la posición. Una prueba de que se la mitigación es que la mayor cantidad de métodos están en la categoría CC y ésta sólo fue dos veces la categoría ganadora, estando en ambos casos sus métodos se ubicaron en los primeros lugares.

Un hecho interesante es que sólo en dos bases de datos hay poca diferencia numérica entre la estrategia ganadora y la segunda, lo que nos hace pensar que hay características de los datos que favorecen a un esquema con respecto al resto.

Según nuestra experiencia previa, una característica de los datos que influye directamente en los resultados de la selección de objetos es el nivel de entremezclado entre las clases. Este tiene una influencia directa tanto en los niveles de reducción alcanzados por los métodos de condensación, como en la cantidad de objetos ruidosos detectados por los métodos de edición. Para validar esta hipótesis fueron realizadas dos acciones:

1. Realización de todos los experimentos en bases de datos sintéticas, de características conocidas. Para esto se generaron 5 bases de datos con 2 clases, 2 atributos numéricos y 1000 objetos por clase, donde cada clase tiene una distribución Gausiana de 2 dimensiones, de desviación estándar 1. En cada caso la primera clase tiene media 0, y la media de la segunda clase fue variada entre 1 y 5 (de ahí el nombre de las bases de datos). Los resultados se muestran en las 5 últimas filas de la Tabla 7

2. Utilización de un meta-dato que permita medir el entremezclado de clases en una base de datos. Los meta-datos (dato calculado a partir de otros datos) han sido utilizados frecuentemente en el reconocimiento de patrones, para medir características de conjuntos de objetos. Ejemplo de ellos son el número de observaciones, medias y desviaciones estándar de los valores de un atributo, número de clases, etc. En algunos casos éstos han sido utilizados para predecir el comportamiento futuro de un método, antes de ser aplicado [85]. En esta tesis se utiliza un meta - dato muy sencillo, pero que permite medir eficazmente cuán entremezcladas están las clases, en un conjunto de objetos dado. Éste se define como la razón de cuántos de los k vecinos más cercanos de cada objeto son de su

misma clase. Estos valores para $k = 7$ pueden verse en las tablas bajo el título *MetaIdx*. Para seleccionar el valor de k se realizaron pruebas para ver su influencia en los resultados, concluyéndose que en un rango de valores alrededor de $k = 7$ se mantienen estables.

Como puede verse claramente en la Tabla 7, en las bases de datos sintéticas, a medida que las clases están menos entremezcladas (el valor de *MetaIdx* aumenta) la estrategia superior pasa de ser CE a EC. Incluso, uniendo los resultados de la Tabla 6 y la Tabla 7, una regla simple explica correctamente 17 de los 20 ejemplos: s*i* $MetaIdx < 0.85$ *la mejor estrategia es EC o CC, si no es CE*.

Con estos resultados previos, se seleccionó el método de edición y el método de condensación que mejores resultados obtuvieron al ser combinados en los dos órdenes. Estos, de acuerdo a la Tabla 4 fueron el CSESupport y el CSEditB, que en 10 de las 15 bases de datos, resultaron estar entre los 5 de mejores resultados. En los epígrafes siguientes se muestran los resultados de la aplicación de estos métodos en todas las bases de datos, combinándose de diferente forma. En todos los casos las bases de datos están ordenadas por el valor de *MetaIdx*, lo que permite ver con más claridad la influencia del entremezclado de las clases en el resultado.

4.6.1 Editar vs Editar+Editar

En éste epígrafe se mostrará la comparación del resultado del método de edición CSEditB con respecto al resultado de dos aplicaciones sucesivas del mismo método.

La Tabla 8 muestra los resultados de la aplicación de los métodos CSEditB y CSEditB + CSEditB, tanto en error como en retención. Dos conclusiones fundamentales pueden ser extraídas:

- La aplicación de un método de edición puede aumentar el error original, en vez de disminuirlo y esto no parece tener relación con el entremezclado. En nuestra opinión esto es debido a que en estos casos no es posible detectar cuándo un objeto es ruidoso, o cuándo simplemente es un objeto poco común en la muestra tomada para entrenar. No obstante, este comportamiento no es común, dándose en sólo 3 de 20 resultados.

- La aplicación de un segundo método de edición puede aumentar el error del primero, o no modificarlo en lo absoluto. El aumento del error se puede apreciar principalmente en presencia de clases muy entremezcladas, ya que el primer método elimina una gran cantidad de objetos, trabajando el segundo con un conjunto con poca relación con el original. La no modificación del resultado se aprecia más en presencia de clases más separadas, pues en éstas hay menos objetos "erróneos", y casi todos son eliminados por el primer método.

Tabla 8. Comparación del resultado de CSEditB con respecto a CSEditB+CSEditB (SCEEditB^2). En negritas, los resultados con menor error por base de datos.

Dis.	*Base de datos*	CSEditB		CSEditB^2		*Error original*	*MetaIdx*
		Error	*% Ret*	*Error*	*% Ret*		
HVDM	**CMC**	**0.536**	0.400	0.540	0.358	0.543	0.510
Eucl	**CMC**	0.467	0.403	**0.464**	0.370	0.540	0.540
Eucl	**GausMean0&1**	0.363	0.585	**0.362**	0.555	0.405	0.631
HVDM	**WPBC**	**0.222**	0.702	**0.222**	0.669	0.247	0.710
Eucl	**WPBC**	0.258	0.657	**0.247**	0.624	0.303	0.710
HVDM	**CylinderBand**	0.265	0.747	0.281	0.733	**0.239**	0.730
Eucl	**GausMean0&2**	0.190	0.754	**0.184**	0.742	0.236	0.784
HVDM	**CreditScr**	**0.151**	0.799	0.154	0.786	0.197	0.830
HVDM	**Phoneme**	0.110	0.891	0.112	0.882	**0.094**	0.860
Eucl	**GausMean0&3**	0.074	0.897	**0.069**	0.892	0.100	0.909
HVDM	**Annealing**	**0.020**	0.946	0.040	0.931	**0.020**	0.920
Eucl	**WDBC**	0.076	0.908	**0.072**	0.906	0.092	0.920
HVDM	**Iris**	**0.047**	0.926	**0.047**	0.926	0.053	0.930
HVDM	**ThyroidDisAnn**	**0.056**	0.928	**0.056**	0.923	0.069	0.940
HVDM	**BreastCWisc**	**0.039**	0.959	**0.039**	0.957	0.044	0.950
Eucl	**Iris**	**0.033**	0.956	**0.033**	0.956	0.040	0.950
HVDM	**WDBC**	0.037	0.945	**0.035**	0.938	0.047	0.950
Eucl	**BreastCWisc**	**0.029**	0.955	0.030	0.954	0.040	0.960
Eucl	**GausMean0&4**	0.026	0.964	**0.024**	0.963	0.035	0.968
Eucl	**GausMean0&5**	**0.005**	0.992	**0.005**	0.991	0.006	0.990

Como conclusión general podemos plantear que aplicar dos ediciones consecutivas no genera usualmente resultados de mayor calidad que la aplicación de una sola.

4.6.2 Condensar vs Condensar + Condensar

En este epígrafe compararemos los resultados de la aplicación del método de condensación CSESupport con respecto a la aplicación del mismo método dos veces, de forma consecutiva. En la Tabla 9 se incluyen, además de los campos de la tabla anterior,

la distancia euclidiana del resultado, ya que en los métodos de condensación, hay que mantener el compromiso de no condensar sacrificando demasiada eficacia.

Tabla 9. Comparación del resultado de CSESupport con respecto a CSESupport + CSESupport (CSESupport ^2). En negrita, los mejores resultados por base de dato.

Dis	*Base de datos*	CSESupport			CSESup^2			*Error original*	*Metaldx*
		Error	*% Ret*	*Dist*	*Error*	*% Ret*	*Dist*		
HVDM	**CMC**	0.557	0.680	0.879	0.566	0.656	**0.866**	0.543	0.510
Eucl	**CMC**	0.561	0.740	0.929	0.568	0.724	**0.920**	0.540	0.540
Eucl	**GausMean0&1**	0.423	0.513	0.665	0.441	0.483	**0.654**	0.405	0.631
HVDM	**WPBC**	0.343	0.427	0.548	0.354	0.376	**0.516**	0.247	0.710
Eucl	**WPBC**	0.333	0.416	0.533	0.353	0.365	**0.508**	0.303	0.710
HVDM	**CylinderBand**	0.315	0.360	0.478	0.348	0.300	**0.459**	0.239	0.730
Eucl	**GausMean0&2**	0.264	0.323	0.417	0.284	0.297	**0.411**	0.236	0.784
HVDM	**CreditScr**	0.245	0.296	**0.384**	0.323	0.227	0.395	0.197	0.830
HVDM	**Phoneme**	0.133	0.185	**0.228**	0.196	0.152	0.248	0.094	0.860
Eucl	**GausMean0&3**	0.118	0.139	**0.182**	0.135	0.127	0.185	0.100	0.909
HVDM	**Annealing**	0.010	0.077	**0.078**	0.110	0.044	0.118	0.020	0.920
Eucl	**WDBC**	0.111	0.129	**0.170**	0.128	0.119	0.175	0.092	0.920
HVDM	**Iris**	0.073	0.119	**0.140**	0.133	0.096	0.164	0.053	0.930
HVDM	**ThyroidDisAnn**	0.110	0.125	**0.167**	0.165	0.091	0.188	0.069	0.940
HVDM	**BreastCWisc**	0.073	0.132	0.151	0.117	0.084	**0.144**	0.044	0.950
Eucl	**Iris**	0.073	0.104	**0.127**	0.120	0.081	0.145	0.040	0.950
HVDM	**WDBC**	0.079	0.099	**0.127**	0.134	0.064	0.148	0.047	0.950
Eucl	**BreastCWisc**	0.060	0.084	**0.103**	0.102	0.064	0.120	0.040	0.960
Eucl	**GausMean0&4**	0.041	0.051	**0.065**	0.063	0.047	0.079	0.035	0.968
Eucl	**GausMean0&5**	0.009	0.014	**0.017**	0.012	0.012	**0.017**	0.006	0.990

Del análisis de la Tabla 9 se pueden extraer las siguientes conclusiones:

- Condensar siempre aumenta el error original. Esto puede deberse a dos factores: ningún objeto es del todo redundante y varios objetos considerados individualmente como redundantes, pueden traer un efecto negativo al ser eliminados en su conjunto
- Condensar dos veces siempre aumenta el error de hacerlo una sola vez. Esto se pude deducir sin mucha dificultad de la conclusión anterior. En algunos casos, sobre todo cuando hay mayor entremezclado entre las clases, este aumento del error puede ser balanceado con una menor retención, y de ahí el menor valor de la distancia

En general, la ganancia en compactación en la aplicación del segundo método de condensación no es muy grande, y se hace a costa de un aumento grande del error, por lo que la estrategia Condensar + Condensar no parece ser recomendable.

4.6.3 Editar vs Editar + Condensar

En este epígrafe se compara la aplicación del método de edición con respecto a la aplicación consecutiva del método de condensación. Se incluye el campo R.G. (razón de ganancia) que expresa la relación entre los dos campos $Dist_{CSEditB} / Dist_{CSEditB+CSESupport}$.

Mientras mayor es este valor, mejor es el resultado de CSEditB + CSESupport en comparación con CSEditB.

Tabla 10. Comparación del resultado de CSEditB con respecto a CSEditB + CSESupport

Dis	*Base de datos*	CSEditB			CSEditB+ CSESupport			*Error Orig.*	*Meta Idx*	*R.G.*
		Error	*% Ret*	*Dist*	*Error*	*% Ret*	*Dist*			
HVDM	**CMC**	0.536	0.400	0.669	0.552	0.138	0.569	0.543	0.510	1.2
Eucl	**CMC**	0.467	0.403	0.617	0.492	0.103	0.503	0.540	0.540	1.2
Eucl	**GausMean0&1**	0.363	0.585	0.688	0.372	0.114	0.389	0.405	0.631	1.8
HVDM	**WPBC**	0.222	0.702	0.736	0.288	0.118	0.311	0.247	0.710	2.4
Eucl	**WPBC**	0.258	0.657	0.706	0.298	0.118	0.321	0.303	0.710	2.2
HVDM	**CylinderBand**	0.265	0.747	0.793	0.300	0.138	0.330	0.239	0.730	2.4
Eucl	**GausMean0&2**	0.190	0.754	0.778	0.200	0.051	0.206	0.236	0.784	3.8
HVDM	**CreditScr**	0.151	0.799	0.813	0.165	0.071	0.180	0.197	0.830	4.5
HVDM	**Phoneme**	0.110	0.891	0.898	0.133	0.080	0.155	0.094	0.860	5.8
Eucl	**GausMean0&3**	0.074	0.897	0.900	0.080	0.020	0.082	0.100	0.909	10.9
HVDM	**Annealing**	0.020	0.946	0.946	0.010	0.060	0.061	0.020	0.920	15.6
Eucl	**WDBC**	0.076	0.908	0.911	0.077	0.021	0.080	0.092	0.920	11.4
HVDM	**Iris**	0.047	0.926	0.927	0.080	0.052	0.095	0.053	0.930	9.7
HVDM	**ThyroidDisAnn**	0.056	0.928	0.930	0.071	0.028	0.076	0.069	0.940	12.2
HVDM	**BreastCWisc**	0.039	0.959	0.960	0.049	0.049	0.069	0.044	0.950	13.9
Eucl	**Iris**	0.033	0.956	0.957	0.033	0.044	0.055	0.040	0.950	17.4
HVDM	**WDBC**	0.037	0.945	0.946	0.063	0.041	0.075	0.047	0.950	12.6
Eucl	**BreastCWisc**	0.029	0.955	0.955	0.036	0.019	0.041	0.040	0.960	23.5
Eucl	**GausMean0&4**	0.026	0.964	0.964	0.028	0.008	0.029	0.035	0.968	33.1
Eucl	**GausMean0&5**	0.005	0.992	0.992	0.007	0.004	0.008	0.006	0.990	123.0

Analizando la Tabla 10, se pueden extraer las siguientes conclusiones:

- Cuando las clases están muy entremezcladas (valores pequeños de *MetaIdx*) , la edición elimina muchos objetos. Esto tiene dos efectos nocivos en el método de

condensación: Pierde el contacto con el problema original, ocasionando un aumento mayor del error y deja menos objetos que eliminar, por lo que condensar pierde bastante de su motivación original.

- Conclusión directa de lo anterior, evidenciado en la tabla: Condensar después de editar tiene un impacto mucho mayor en el resultado, a medida que las clases están más separadas (ver valores del campo *R.G.*)

- En la mayor parte de las bases de datos se aprecia el funcionamiento sinérgico, ya que la edición inicial disminuye el error, y el método de condensación lo aumenta, pero a un nivel inferior al original

Como conclusión final tenemos que Editar + Condensar aprovecha eficazmente la sinergia entre los métodos, en bases de datos con poco nivel de entremezclado.

4.6.4 Condensar vs Condensar + Editar

A continuación compararemos el comportamiento del método de condensación, con respecto a la aplicación posterior del método de edición.

De los resultados de la Tabla 11 podemos extraer las siguientes conclusiones:

- El comportamiento sinérgico de la combinación se aprecia al observar que condensar elimina inicialmente objetos aumentando el error, y posteriormente editar reduce el error, dando resultados de más calidad

- A medida que las clases se separan más, la aplicación del método de edición no sólo no logra reducir el error original, sino hasta puede aumentar el error de condensar. La explicación de este comportamiento se debe a que con clases muy separadas el método de condensación elimina muchos objetos, perdiendo el método de edición contacto con el problema original.

Como conclusión general de este epígrafe se tiene que el comportamiento sinérgico entre Condensar y Editar se logra en bases de datos donde las clases tengan altos niveles de entremezclado, no siendo así en las que tiene clases bien separadas.

Tabla 11. Comparación del resultado de CSESupport con respecto a CSESupport + CSEditB. En negrita, los resultados con menor error por bases de datos.

Dist	*Base de datos*	CSESupport			CSESupport + CSEditB			*Error Orig.*	*MetaIdx*
		Error	*% Ret*	*Dist*	*Error*	*% Ret*	*Dist*		
HVDM	**CMC**	0.557	0.680	0.879	**0.540**	0.030	0.541	0.543	0.510
Eucl	**CMC**	0.561	0.740	0.929	**0.500**	0.090	0.508	0.540	0.540
Eucl	**GausMean0&1**	0.423	0.513	0.665	**0.337**	0.027	0.338	0.405	0.631
HVDM	**WPBC**	0.343	0.427	0.548	**0.222**	0.051	0.228	0.247	0.710
Eucl	**WPBC**	0.333	0.416	0.533	**0.232**	0.045	0.236	0.303	0.710
HVDM	**CylinderBand**	0.315	0.360	0.478	0.344	0.056	0.349	**0.239**	0.730
Eucl	**GausMean0&2**	0.264	0.323	0.417	**0.171**	0.026	0.173	0.236	0.784
HVDM	**CreditScr**	0.245	0.296	0.384	**0.159**	0.068	0.173	0.197	0.830
HVDM	**Phoneme**	0.133	0.185	0.228	0.192	0.031	0.194	**0.094**	0.860
Eucl	**GausMean0&3**	0.118	0.139	0.182	**0.080**	0.013	0.081	0.100	0.909
HVDM	**Annealing**	0.010	0.077	0.078	0.070	0.026	0.075	**0.020**	0.920
Eucl	**WDBC**	0.111	0.129	0.170	0.207	0.012	0.207	**0.092**	0.920
HVDM	**Iris**	0.073	0.119	0.140	0.633	0.022	0.633	**0.053**	0.930
HVDM	**ThyroidDisAnn**	0.110	0.125	0.167	0.070	0.036	0.079	**0.069**	0.940
HVDM	**BreastCWisc**	0.073	0.132	0.151	**0.043**	0.076	0.087	0.044	0.950
Eucl	**Iris**	0.073	0.104	0.127	0.547	0.022	0.547	**0.040**	0.950
HVDM	**WDBC**	0.079	0.099	0.127	0.074	0.039	0.084	**0.047**	0.950
Eucl	**BreastCWisc**	0.060	0.084	0.103	0.049	0.025	0.055	**0.040**	0.960
Eucl	**GausMean0&4**	0.041	0.051	0.065	0.038	0.005	0.038	**0.035**	0.968
Eucl	**GausMean0&5**	0.009	0.014	0.017	0.058	0.002	0.058	**0.006**	0.990

4.6.5 Editar + Condensar vs Condensar + Editar

De las conclusiones de los epígrafes anteriores se puede extraer ya una explicación sobre la influencia del entremezclado de las clases con respecto al orden óptimo de aplicación de los métodos de selección de objetos. No obstante, en este epígrafe se presenta la comparación directa entre editar + condensar y condensar + editar.

De los resultados de la Tabla 12 se puede concluir que la estrategia de Editar + Condensar es superior en las bases de datos con poco entremezclado de clases, mientras que la estrategia Condensar + Editar es superior en caso contrario. Como puede apreciarse mientras menos entremezcladas están las clases, el método de edición reduce menos el error original, y hasta llega a aumentarlo. Sin embargo, cuando las clases están muy entremezcladas, la edición elimina demasiados objetos, por lo que el error da valores mucho mayores que la otra estrategia.

Tabla 12. Comparación del resultado de CSEditB + CSESupport con respecto a CSESupport + CSEditB. En negrita, los resultados con menor distancia.

Dis	*Base de datos*	CSEditB + CSESupport			CSESupport + CSEditB			*Error Original*	*MetaIdx*
		Error	*% Ret*	*Dist*	*Error*	*% Ret*	*Dist*		
HVDM	**CMC**	0.552	0.138	0.569	0.540	0.030	**0.541**	0.543	0.510
Eucl	**CMC**	0.492	0.103	**0.503**	0.500	0.090	0.508	0.540	0.540
Eucl	**GausMean0&1**	0.372	0.114	0.389	0.337	0.027	**0.338**	0.405	0.631
HVDM	**WPBC**	0.288	0.118	0.311	0.222	0.051	**0.228**	0.247	0.710
Eucl	**WPBC**	0.298	0.118	0.321	0.232	0.045	**0.236**	0.303	0.710
HVDM	**CylinderBand**	0.300	0.138	**0.330**	0.344	0.056	0.349	0.239	0.730
Eucl	**GausMean0&2**	0.200	0.051	0.206	0.171	0.026	**0.173**	0.236	0.784
HVDM	**CreditScr**	0.165	0.071	0.180	0.159	0.068	**0.173**	0.197	0.830
HVDM	**Phoneme**	0.133	0.080	**0.155**	0.192	0.031	0.194	0.094	0.860
Eucl	**GausMean0&3**	0.080	0.020	0.082	0.080	0.013	**0.081**	0.100	0.909
HVDM	**Annealing**	0.010	0.060	**0.061**	0.070	0.026	0.075	0.020	0.920
Eucl	**WDBC**	0.077	0.021	**0.080**	0.207	0.012	0.207	0.092	0.920
HVDM	**Iris**	0.080	0.052	**0.095**	0.633	0.022	0.633	0.053	0.930
HVDM	**ThyroidDisAnn**	0.071	0.028	**0.076**	0.070	0.036	0.079	0.069	0.940
HVDM	**BreastCWisc**	0.049	0.049	**0.069**	0.043	0.076	0.087	0.044	0.950
Eucl	**Iris**	0.033	0.044	**0.055**	0.547	0.022	0.547	0.040	0.950
HVDM	**WDBC**	0.063	0.041	**0.075**	0.074	0.039	0.084	0.047	0.950
Eucl	**BreastCWisc**	0.036	0.019	**0.041**	0.049	0.025	0.055	0.040	0.960
Eucl	**GausMean0&4**	0.028	0.008	**0.029**	0.038	0.005	0.038	0.035	0.968
Eucl	**GausMean0&5**	0.007	0.004	**0.008**	0.058	0.002	0.058	0.006	0.990

Por otra parte el comportamiento sinérgico de la estrategia Condensar + Editar se aprecia al observar que condensar elimina inicialmente objetos aumentando el error, y posteriormente editar reduce el error, dando resultados de más calidad. La estrategia Editar + Condensar en la mayor parte de las bases de datos funciona sinérgicamente, ya que la edición inicial disminuye el error, y el método de condensación lo aumenta, pero a un nivel inferior al original

Es por estas razones que concluimos que el comportamiento sinérgico de una estrategia depende fuertemente de la estructura interna de los datos, siendo el entremezclado entre las clases un factor muy importante.

5 Conclusiones y Trabajo Futuro

5.1 Conclusiones

La selección de objetos de entrenamiento es un proceso con un impacto importante en la calidad de los clasificadores supervisados, sobre todo en aquellos que no utilizan una etapa de pre-procesamiento de los datos de entrada, como es el caso del kNN. Por este motivo ésta es un área muy explorada, a la que se le han dedicado una cantidad enorme de artículos científicos, y donde se han creado cientos de métodos y variantes. Recientemente se ha comenzado a prestar más atención a la combinación de métodos, que aprovechen de manera sinérgica las propiedades de los métodos constituyentes de la combinación. Este hecho se ha reflejado de dos formas: con la creación de nuevos métodos que son combinaciones, y con estudios comparativos sobre las diferentes formas de realizar estas combinaciones.

En este trabajo se crearon dos nuevos métodos de condensación y dos nuevos métodos de edición, con un comportamiento individual satisfactorio y con buenos resultados al ser combinados, tanto entre ellos, como con otros previamente reportados. Los nuevos métodos funcionaron eficaz y eficientemente con datos mezclados e incompletos. Todos estos métodos mostraron estar al nivel de los del estado del arte de manera individual, y generaron una mejora del estado del arte utilizados en combinaciones, entre ellos, y con el resto. Las evaluaciones se realizaron utilizando la distancia euclidiana entre el par (% de retención, error) y el punto (0,0) y utilizando la alteración de la frontera de Pareto.

Una amplia experimentación con bases de datos reales y sintéticas, con dos funciones de similaridad diferentes, y la aplicación de 13 métodos, incluyendo los propuestos en este trabajo permitió llegar a algunas conclusiones interesantes.

En primer lugar los métodos individuales dieron siempre resultados inferiores a las combinaciones. Esto muestra que independientemente de la calidad individual de un método, siempre es mejorable si se le combina adecuadamente con otro.

En segundo lugar, en casi todas las bases de datos una de las estrategias fue marcadamente superior a las demás y esto tiene una relación directa con el índice de entremezclado de sus clases. Este entremezclado, medido con un meta - dato apropiado, fue capaz de identificar en 17 de 20 bases de datos cuál era la estrategia ganadora.

Un análisis posterior de los resultados de la aplicación de un método de edición y otro de condensación en todas las bases de datos, permitió llegar a las siguientes conclusiones:

- No siempre la aplicación de un método de edición disminuye el error original y, en general, la aplicación de un segundo método de edición sobre el resultado del primero no da resultados superiores.
- La aplicación de un método de condensación siempre aumenta el error original. La aplicación de un segundo método de condensación sobre el resultado del primero tiene un efecto negativo en el resultado final, sobre todo si las clases están poco entremezcladas.
- Editar + Condensar muestra resultados sinérgicos de más calidad en las bases de datos de clases poco entremezcladas. Por el contrario, en las que lo están, el método de edición elimina demasiados objetos, trabajando el método de condensación con un problema muy diferente al original, con malos resultados.
- Condensar + Editar muestra resultados sinérgicos superiores en las bases de datos con mayor entremezclado. En cambio, con niveles bajos de entremezclado, el método de edición no disminuye el error, sino más bien tiende a aumentarlo, debido a la eliminación de demasiados objetos por el método de condensación en estas condiciones.

5.2 Principales Aportaciones

El presente trabajo tiene cinco aportaciones fundamentales.

En primer lugar la creación de dos nuevos métodos de condensación y dos nuevos métodos de edición, que mostraron en una amplia experimentación estar individualmente al nivel de los del estado del arte. Al ser combinados entre ellos y con otros métodos reportados previamente, producen una mejora apreciable del estado del arte en la materia.

En segundo lugar se realizó una amplia experimentación con 13 métodos en total, individuales y combinados en todas las parejas posibles, en 10 bases de datos reales y 5 sintéticas, utilizando dos funciones de comparación diferentes muy populares en la literatura del área. Un estudio de esta extensión, que hasta donde tenemos conocimiento no se ha realizado antes, permitió disponer de datos para llegar a conclusiones sobre el funcionamiento de las combinaciones y la influencia del orden en la calidad de los resultados. Además, para la valoración de los resultados se utilizaron fronteras de Pareto, lo que ha sido poco utilizado en trabajos previos, a pesar de sus ventajas en problemas multi-objetivos.

En tercer lugar se mostró que los resultados de la aplicación de todos los métodos de manera individual fueron superados por alguna combinación de métodos. Esto es un elemento que no obstante lo bueno que puede resultar un método, siempre se pueden obtener resultados superiores si se le combina adecuadamente.

En cuarto lugar se mostró la fuerte dependencia del entremezclado de las clases en el orden más favorable de aplicación de los métodos. Para esto se utilizó un metadato que calcula eficaz y eficientemente este nivel de entremezclado.

Finalmente se mostró el efecto de varias estrategias de combinación de métodos sobre el resultado final de la selección de objetos. Se expuso el por qué algunas estrategias son superiores que otras y la dependencia que esto tiene con el entremezclado de las clases. Se mostró, además, que las estrategias de Editar + Editar y Condensar + Condensar no generan buenos resultados, mientras que Editar + Condensar es la mejor estrategia para bases de datos con poco entremezclado de clases y Condensar + Editar es la mejor en las de mayor entremezclado. Estos resultados pueden ser de gran utilidad práctica dada la

cantidad tan grande de métodos de selección posibles a aplicar, y por tanto, el gran volumen de combinaciones que de ellos pudieran obtenerse.

5.3 Trabajos Futuros

Aunque el entremezclado de las clases, medido con el meta - dato propuesto, explica una parte importante de los resultados experimentales obtenidos, existen otros factores a tomar en cuenta para seleccionar de antemano la mejor estrategia a seguir para seleccionar objetos. El tipo de los atributos, el balance entre las clases, la cantidad de datos incompletos son algunos ejemplos de factores de los que se puede investigar su influencia en la calidad de los resultados. Este es un campo muy poco explorado y que amerita continuar las investigaciones.

Por otra parte la selección de qué método (o métodos) a combinar, y cómo, basándose en las características de una base de datos en particular, es un tema que se ha trabajado aún en menor medida que el anterior. Muchos trabajos se han limitado a comparaciones experimentales donde muestran que los nuevos métodos propuestos funcionan de manera superior a los anteriores, en algunas bases de datos en particular. Como resultado se dispone en la actualidad de cientos de métodos y variantes, pero dado un problema en particular, no se tiene otra estrategia que no sea la prueba de todos los métodos disponibles, y la selección del mejor. Es por esto y por la importancia de la buena selección de los objetos de entrenamiento para prácticamente cualquier clasificador, que este tema nos parece de vital importancia.

Referencias Bibliográficas

1. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification. IEEE Trans. on Information Theory **IT-13** (1967) 21-27
2. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2nd Edition). Wiley-Interscience (2000)
3. Bezdek, J.C., Kuncheva, L.I.: Nearest Prototype classifiers design: an experimental study. International Journal of Intelligent Systems **16** (2001) 1445-1473
4. Kim, S.-W., Oommen, J.B.: A brief taxonomy and ranking of creative prototype reduction schemes. Pattern Analysis and Applications **6** (2003) 232-244
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, J.: Pattern recognition: a statistical approach. Prentice/Hall International, Englewood Cliffs, N.J. (1982)
7. Hart, P.E.: The condensed nearest neighbor rule. IEEE Trans. on Information Theory **14** (1968) 515-516
8. Kuncheva, L.I., Bezdek, J.C.: Nearest prototype classification: clustering, genetic algorithms or random search. IEEE transactions on systems, man and cybernetics. Part C **28** (1998) 160-164
9. Gates, G.W.: The reduced nearest neighbor rule. IEEE Trans. on Information Theory **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. **24** (1994) 511-517
11. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on systems, man and cybernetics **SMC-2** (1972) 408-421
12. Tomek, I.: An experiment with the Edited Nearest-Neighbor Rule. IEEE Transactions on Systems, Man and Cybernetics **SMC-6** (1976) 448-452
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** (2003) 1015-1022
14. Wang, C., Chen, Y.Q.: Improving Nearest Neighbor classification with simulated gravitational collapse. Advances in Natural Computation, Pt 3, Proceedings **3612** (2005) 845-854
15. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery **6** (2002) 153-172
16. Dasarathy, B.V., Sanchez, J.S., Townsend, S.: Nearest Neighbor Editing and Condensing Tools - Synergy Exploitation. Pattern Analysis & Applications **3** (2000) 19-30
17. Devijver, P.A., Kittler, J.: On the edited nearest neighbor rule. In: Press, I.C.S. (ed.): 5th International Conference on Pattern Recognition, Los Alamitos, California (1980) 72-80
18. Charu, C.A.: On the effects of dimensionality reduction on high dimensional similarity search. Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM Press, Santa Barbara, California, United States (2001)

19. Kuncheva, L.I.: Editing for the k-nearest neighbors rule by a genetic algorithm. Pattern Recognition Letters (1995)
20. Sanchez, J.S., Pla, F., Ferri, F.J.: Prototype selection for the nearest neighbor rule through proximity graphs. Pattern Recognition Letters **18** (1997) 507-513
21. Li, Y., Huang, J., Zhang, W., Zhang, X.: New prototype selection rule integrated condensing with editing process for the nearest neighbor rules. IEEE International Conference on Industrial Technology. ICIT 2005 (2005) 950-954
22. Wilson, R.D., Martinez, T.R.: Reduction techniques for Instance-based learning algorithms. Machine Learning **38** (2000) 257-286
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 (1998)
24. Gómez-Herrera, J.E.: Prognostic of gas-petroleum in the Cuban ophiolitic rocks association, applying mathematical modeling. Geofísica Internacional **33** (1994) 447-467
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 in Computer Science **1910** (2000) 524-529
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 (2002) 133-176
27. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and Unsupervised Discretization of Continuous Features. 12th International Conference on Machine Learning (1995) 194-202
28. Lumijarvi, J., Laurikkala, J., Juhola, M.: A comparison of different heterogeneous proximity functions and Euclidean distance. Medinfo **11** (2004) 1362-1366
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** (1992) 159-171
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 (1993) 1-61
31. Duch, W., Grudzinski, K., G., S.: Symbolic Features in Neural Networks. 5th Conference on Neural Networks and Soft Computing, Zakopane (2000) 180-185
32. Stanfill, C., Walts, D.L.: Toward memory-based reasoning. Communications of the ACM **29** (1986) 1213-1228
33. Maudes, J., Rodríguez, J.J., García-Osorio, C.: Cascading for nominal data. Lecture Notes in Computer Science **4472** (2007) 231-240
34. Wilson, R.D., Martinez, T.R.: Improved Heterogeneous Distance Functions. Journal of Artificial Intelligence Research **6** (1997) 1-34
35. Pekalska, E., Paclík, P., Duin, R.P.W.: A generalized kernel approach to dissimilarity-based classification. Journal of Machine Learning Research **2** (2001) 175-211
36. Pekalska, E., Duin, R.P.W.: Automatic pattern recognition by similarity representations. IEEE Electronic Letters **32** (2001) 159-160

37. Martínez-Trinidad, J.F., Guzmán-Arenas, A.: The logical combinatorial approach to Pattern Recognition, an overview through selected works. Pattern Recognition **34** (2001) 741-751
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** (2002) 147-177
41. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning **6** (1991) 37-66
42. Ventura, D., Martínez, T.R.: An empirical comparison of discretization methods. Tenth International Symposium on Computer and Information Sciences (1995) 443-450
43. Ullmann, J.R.: Automatic selection of reference data for use in a nearest neighbor method of pattern classification. IEEE Transactions on Information Theory **20** (1974) 541-543
44. Tomek, I.: Two modifications of CNN. IEEE Transactions on systems, man and cybernetics **SMC-6** (1976) 769-772
45. Kibler, D., Aha, D.W.: Learning representative exemplars of concepts: An initial case study. Fourth international workshop on Machine learning, Irvine, CA (1987) 24-30
46. Angiulli, F.: Fast condensed nearest neighbor rule. Proceedings of the 22nd international conference on Machine learning. ACM Press, Bonn, Germany (2005)
47. 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 (1972) 511-519
48. Ritter, G.L., Woodruff, H.B., Lowry, S.R., Isenhour, T.L.: An algorithm for a selective nearest neighbor decision rule. IEEE Trans. on Information Theory **21** (1975) 665-669
49. Zhang, H., Sun, G.: Optimal reference subset selection for nearest neighbor classification by tabu search. Pattern Recognition **35** (2002) 1481-1490
50. Dasarathy, B.V.: Nearest unlike neighbor (NUN): an aid to decision confidence estimation. Optical Engineering **34** (1995) 2785-2792
51. Wilfong, G.: Nearest neighbor problems. 7th Annual ACM Sumposium on Computational Geometry (1991) 224-233
52. Jaromczyk, J.W., Toussaint, G.T.: Relative Neighborhood Graphs and their relatives. Proceedings of IEEE (1992)
53. Toussaint, G.T.: Proximity Graphs for Nearest Neighbor Decision Rules: Recent Progress. 34 Symposium on Computing and Statistics INTERFACE-2002, Montreal, Canada (2002)
54. Bhattacharya, B.K., Poulsen, R.S., Toussaint, G.T.: Application of Proximity Graphs to Editing Nearest Neighbor Decision Rules. 16th Symposium on the Interface Between Computer and Human (1984)
55. Delaunay, B.: Sur la sphère vide. Bulletin Academy Science USSR **7** (1934) 793-800

56. Takekazu, K., Toshikazu, W.: Direct Condensing: An Efficient Voronoi Condensing Algorithm for Nearest Neighbor Classifiers. Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03. IEEE Computer Society (2004)
57. Barber, C.B., David, P.D., Hannu, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22** (1996) 469-483
58. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience (2004)
59. Chang, I.E., Lippman, R.P.: Using genetic algorithms to improve pattern classification performance. Advances in neural information processing **3** (1991) 797-803
60. Skalak, D.B.: Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. Eleventh International Conference on Machine Learning (1994) 293-301
61. Mitchell, M., Holland, J.H., Forrest, S. (eds.): When will a genetic algorithm outperform Hill - Climbing. Morgan Kaufmann, San Mateo, CA (1994)
62. Zhang, J.: Selecting Typical Instances in Instance-based learning. Ninth International Machine Learning Workshop. Morgan Kaufmann, San Mateo, Ca., Aberdeen, Scotland (1992) 470-479
63. Wilson, R.D.: Advances in Instance-Based learning algorithms. Department of Computer Science of Brigham Young University, Brigham (1997)
64. Fred, G., Fred, L.: Tabu Search. Kluwer Academic Publishers (1997)
65. Huang, D., Chow, T.W.S.: Enhancing density-based data reduction using entropy. Neural Computation **18** (2006) 470-495
66. García-Borroto, M., Ruiz-Shulcloper, J.: Selecting Prototypes in Mixed Incomplete Data. Lecture Notes in Computer Science **3773** (2005) 450-459
67. Martínez-Trinidad, J.F., Ruiz-Shulcloper, J., Lazo-Cortés, M.S.: Structuralization of universes. Fuzzy sets and systems **112** (2000) 485-500
68. 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** (2005) 528-535
69. Huang, C.C.: A novel gray-based reduced NN classification method. Pattern Recognition **39** (2006) 1979-1986
70. Deng, J.L.: Introduction to Grey system theory. J. Grey Syst. **1** (1989) 1-24
71. Fayed, H.A., Hashem, S.R., Atiya, A.F.: Self-generating prototypes for pattern classification. Pattern Recognition **40** (2007) 1498-1509
72. Davison, A.C., Hinkley, D.V.: Bootstrap Methods and their Application. Cambridge Press (1997)
73. Chang, C.L.: Finding prototypes for nearest neighbor classifier. IEEE transactions on computers **23** (1974) 1179-1184
74. 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 (2002)
75. Bezdek, J.C., Reichherzer, T.R., Lim, G.S., Y., A.: Multiple-prototype classifier design. IEEE transactions on systems, man and cybernetics. Part C **28** (1998) 67-79
76. Kohonen, T.: Self - Organizing Maps. Springer, Germany (1995)

77. Geva, S., Sitte, J.: Adaptive nearest neighbor pattern classification. IEEE transactions on neural networks **2** (1991) 318-322
78. Odorico, R.: Learning vector quantization with training counters (LVQTC). Neural Newtorks **10** (1997) 1083-1088
79. Xie, Q., Laszlo, C.A., Ward, R.K.: Vector Quantization Technique for Nonparametric Classifier Design. IEEE Trans. Pattern Anal. Mach. Intell. **15** (1993) 1326-1330
80. Hamamoto, Y., Uchimura, S., Tomita, S.: A bootstrap technique for nearest neighbor classifier design. IEEE transactions on pattern analysis and Machine intelligence **19** (1997) 73-79
81. Hattori, K., Takahashi, M.: A new edited k-nearest neighbor rule in the pattern clasification problem. Pattern Recognition **33** (2000) 521-528
82. Brighton, H., Mellish, C.: On the Consistency of Information Filters for Lazy Learning Algorithms. Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery. Springer-Verlag (1999)
83. Gil-García, R., Badia-Contelles, J.M., Pons-Porrata, A.: Dynamic hierarchical compact clustering algorithm. Progress in Pattern Recognition, Image Analysis and Applications, Proceedings **3773** (2005) 302-310
84. Cohon, J.: Multiobjective Programming and Planning. John Wiley, New York (1978)
85. Sohn, S.Y.: Meta Analysis of Classification Algorithms for Pattern Recognition. IEEE Trans. Pattern Anal. Mach. Intell. **21** (1999) 1137-1144

Anexo 1. Pseudocódigos de Algoritmos de Selección utilizados en las comparaciones

Condensed Nearest Neighbors (CNN)

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	$V \Leftarrow GrabBag[0]$	**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 $V \Leftarrow y$ $GrabBag$ $moved \leftarrow true$	**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	

Minimal Consistent Subset (MCS)

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 $x \in T$ $dN \leftarrow d(x, NUN(x, Current))$ $Nearest_x \leftarrow \{y \in T : d(y,x) < dN\}$	**Para cada muestra, hallar la distancia al NUN y las muestras de su soporte**
	foreach $y \in Nearest_x$ $add(Voters_y, x)$	**Para cada una de las muestras que se encontraban más cerca que el NUN se actualiza su conjunto de votantes**
	if $Nearest_x = \{x\}$ then $V \leftarrow V \cup \{x\}$	**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	$V \leftarrow V \cup \{mvs\}$	
7	foreach $z \in Voters_{mvs}$ foreach $q \in T$ $Voters_q \leftarrow Voters_q \setminus \{z\}$	**Se actualizará 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 $\lvert Curren \rvert t \neq \lvert V \rvert$ then $Current \leftarrow V$ goto 2	**Se inicia una nueva iteración, si se han eliminado algún objeto en la iteración actual**
10	end	

Decremental reduction optimization procedures (DROP – 3)

Notación y Definiciones:

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

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	$NewT \leftarrow ENN(T)$	**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 vecino de x** **x es entonces, asociado de y**
5	$sortAsc(V, \prec)$	
6	foreach $v \in V$ $with \leftarrow \varepsilon_V^{kNN}(Associated_v)$ $without \leftarrow \varepsilon_{V \setminus \{v\}}^{kNN}(Associated_v)$	**Error al clasificar los asociados, eliminando a** v **de la matriz de entrenamiento del *kNN***
	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 $y \in NN_x$ $Associated_y \leftarrow Associated_y \cup x$	**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	

Typical instance – based learning (TIBL)

Notación y definiciones:

$$d(x,y)=\sqrt{\frac{1}{n}\sum_{r_i\in R}\left(\frac{r_i(x)-r_i(y)}{\max_{z\in T}\{r_i(z)\}-\min_{z\in T}\{r_i(z)\}}\right)^2}$$ **Distancia utilizada en el algoritmo**

Entradas:

T **Matriz de entrenamiento**

Salida:

V **Resultado del método**

Pasos:

1	foreach x *in* T	**Para cada elemento de la matriz de entrenamiento**
	$Cx \leftarrow \{y \in T \setminus \{x\} : \alpha(y)=\alpha(x)\}$	**se buscan las instancias pertenecientes a su clase**
	$Cnx \leftarrow \{y \in T : \alpha(y)\neq\alpha(x)\}$	**y las que no pertenecen a su clase**
	$typicality_x \leftarrow \frac{\frac{1}{\lvert Cnx\rvert}\sum_{y\in Cnx} d(x,y)}{\frac{1}{\lvert Cx\rvert}\sum_{y\in Cx} d(x,y)}$	**Función que devuelve 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 $\varepsilon_V^{kNN}(T)\neq 0$ then goto **3**	**Si existen errores de clasificación se repite el proceso**
10	end	

Compact Set Editing (CSE)

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**
$S(x)$	$\{b \in Vert : (x,b) \in C\}$**, conjunto de sucesores del vértice *x* en el grafo. La presencia de uno de estos elementos de la misma clase de x en $Vert$ garantiza la correcta clasificación de x (si se utiliza un clasificador 1-NN). Lo llamaremos** *soporte* **de x**
$A(x)$	$\{a \in Vert : (a,x) \in C\}$**, conjunto de antecesores del vértice x en el grafo**
$x \prec y$	$x \prec y \Leftrightarrow (e_x < e_y) \vee (e_x = e_y \wedge s_x < s_y) \vee (e_x = e_y \wedge s_x = s_y \wedge \lvert Flags_x \rvert > \lvert Flags_y \rvert)$ **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**

Salida:

V	**Resultado del método**

Entradas:

T	**Matriz de entrenamiento**

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 quiénes 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 $y \in S(x)$ $Flags_y \leftarrow Flags_y \cup \{x\}$	**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 $Flags_y$ $Flags_y \leftarrow Flags_y \setminus \{x\}$	**Como el objeto x va a ser eliminado, ya no puede soportar a ningún otro objeto, y por tanto se elimina de todos los $Flags$**
3	foreach $y \in S(x)$ $e_y \leftarrow e_y - 1$	**Los sucesores tienen un antecesor menos**
4	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)**
5	$T \leftarrow T \setminus \{x\}$	**Se elimina x y sus aristas asociadas**
6	$C \leftarrow C \setminus \{(a,b) \in C : a = x \vee b = x\}$	

Edited Nearest Neighbors (ENN)

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 $Bad \leftarrow Bad \bigcup \{x\}$	**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	

Anexo 2. Fronteras de Pareto

En éste apéndice aparecen las fronteras de Pareto antes y después de la inclusión de las combinaciones con los métodos propuestos en este trabajo. Aparecen organizadas alfabéticamente por el nombre de la base de datos, incluyéndose consecutivamente los resultados con las dos funciones de comparación, para las bases de datos numéricas.

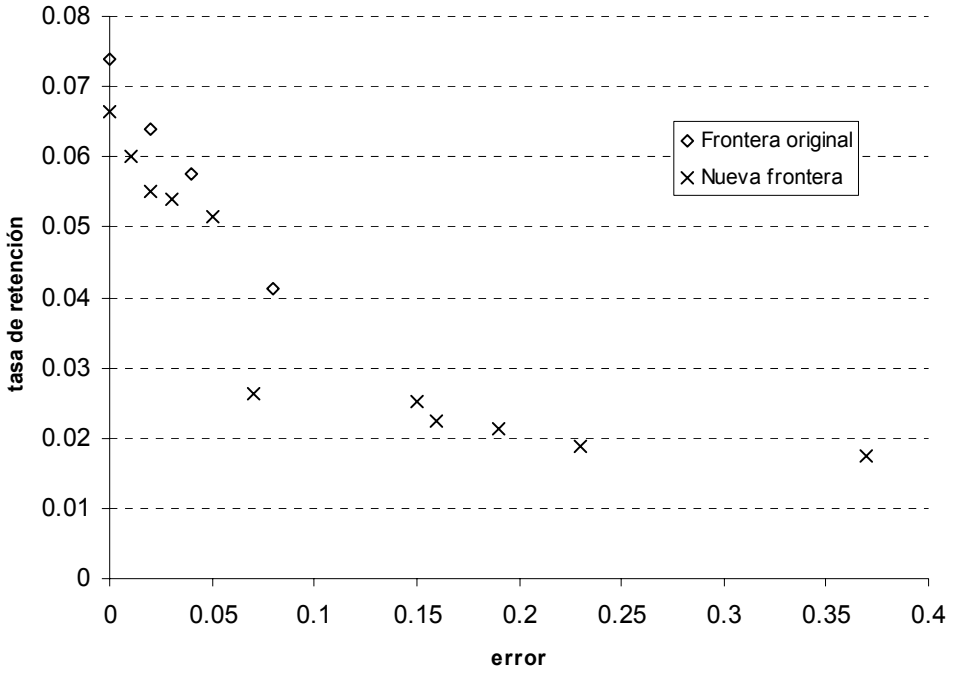


Figura 20. Fronteras de Pareto en base de datos Annealing (HVDM), antes y después de incluir los nuevos métodos

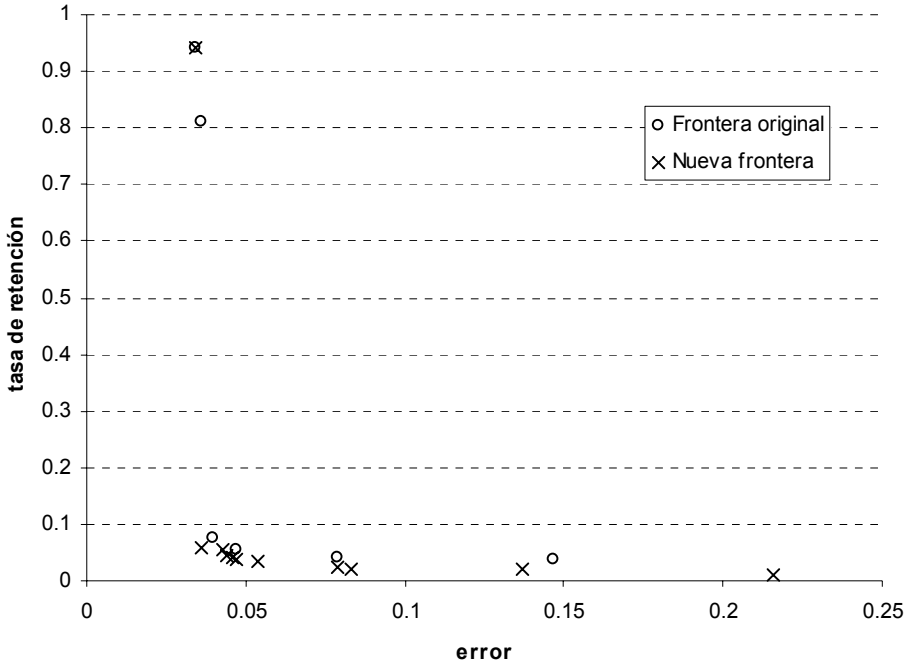


Figura 21. Fronteras de Pareto en base de datos BreastCancerWisconsing (HVDM), antes y después de incluir los nuevos métodos

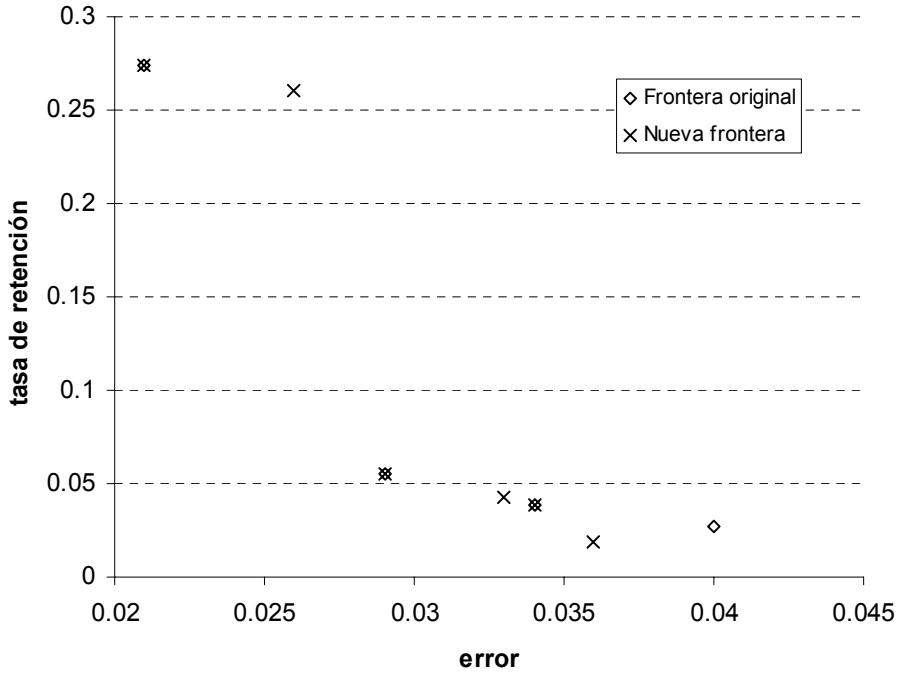


Figura 22. Fronteras de Pareto en base de datos BreastCancerWisconsing (Euclidean), antes y después de incluir los nuevos métodos

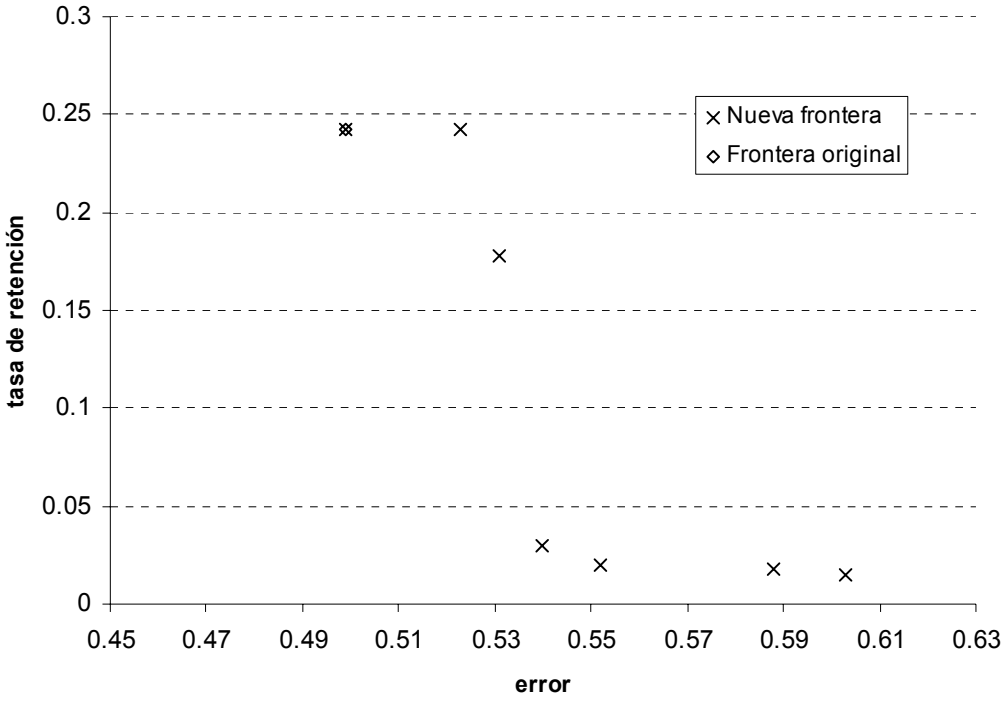


Figura 23. Fronteras de Pareto en base de datos CMC (HVDM), antes y después de incluir los nuevos métodos

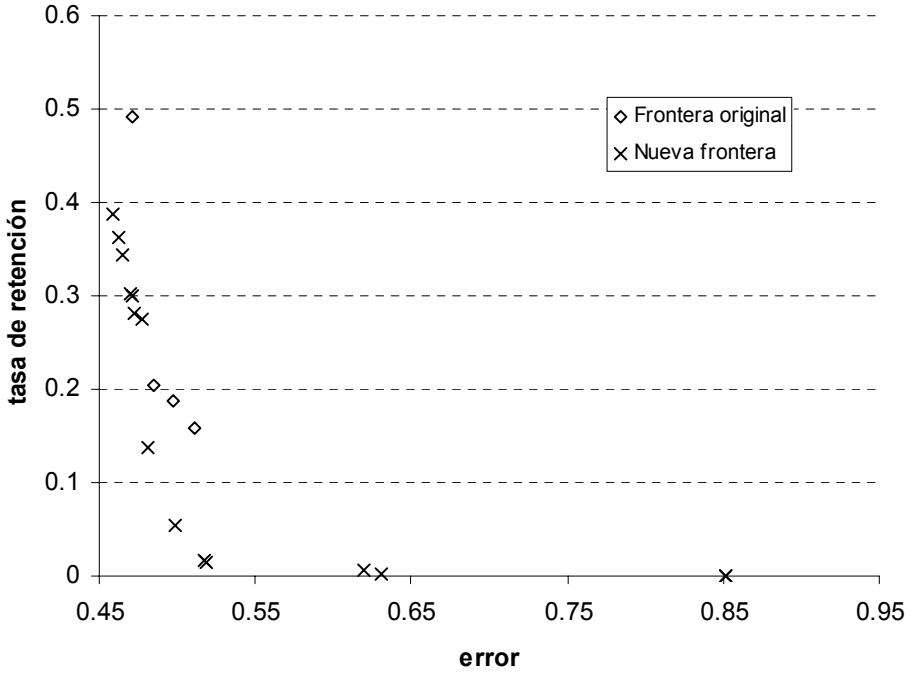


Figura 24. Fronteras de Pareto en base de datos CMC (Euclidean), antes y después de incluir los nuevos métodos

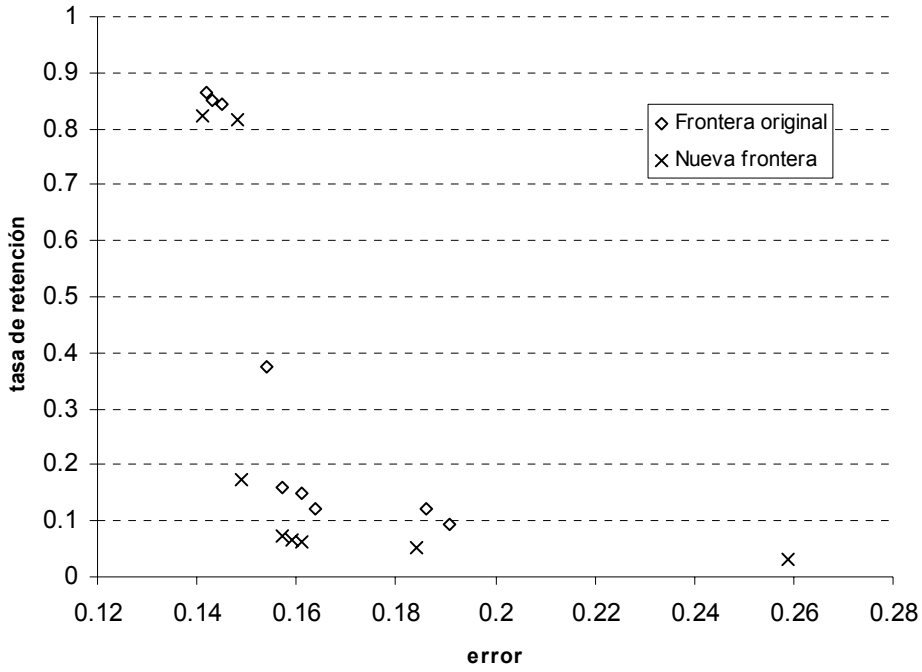


Figura 25. Fronteras de Pareto en base de datos CreditScreening (HVDM), antes y después de incluir los nuevos métodos

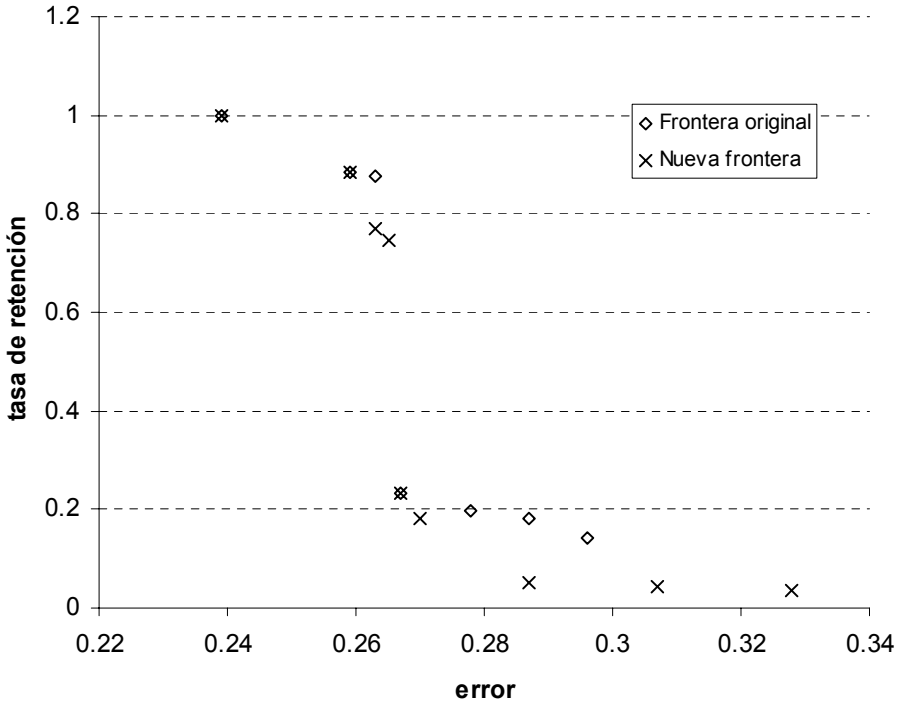


Figura 26. Fronteras de Pareto en base de datos CylinderBand (HVDM), antes y después de incluir los nuevos métodos

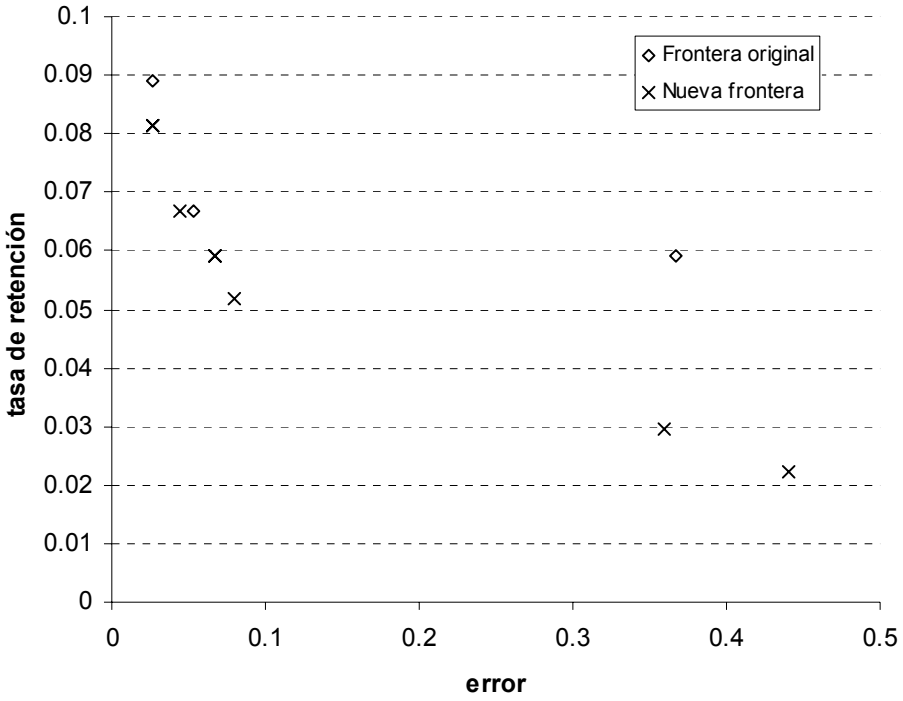


Figura 27. Fronteras de Pareto en base de datos Iris (HVDM), antes y después de incluir los nuevos métodos

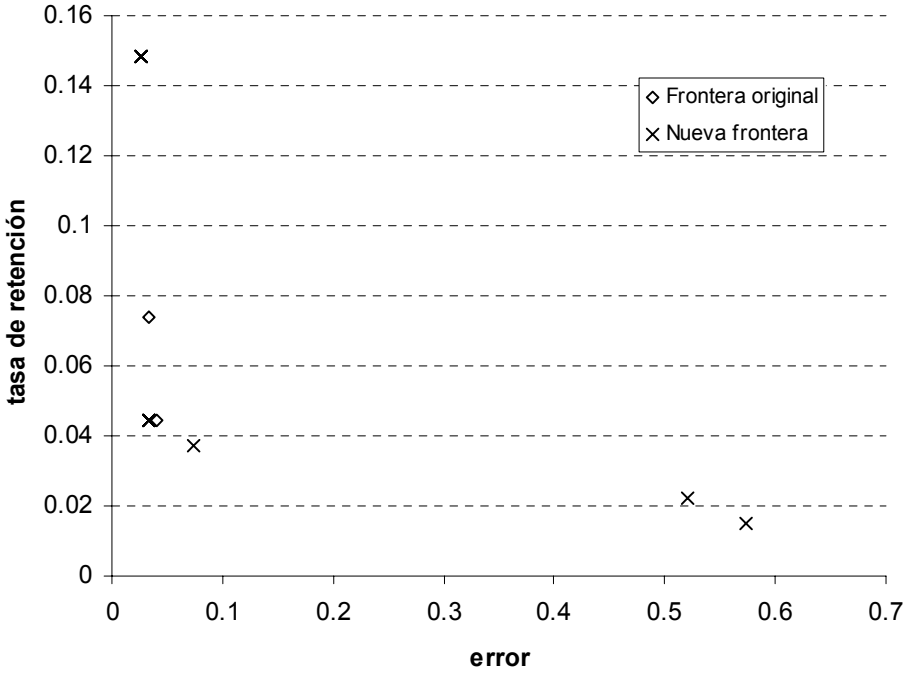


Figura 28. Fronteras de Pareto en base de datos Iris (Euclidean), antes y después de incluir los nuevos métodos

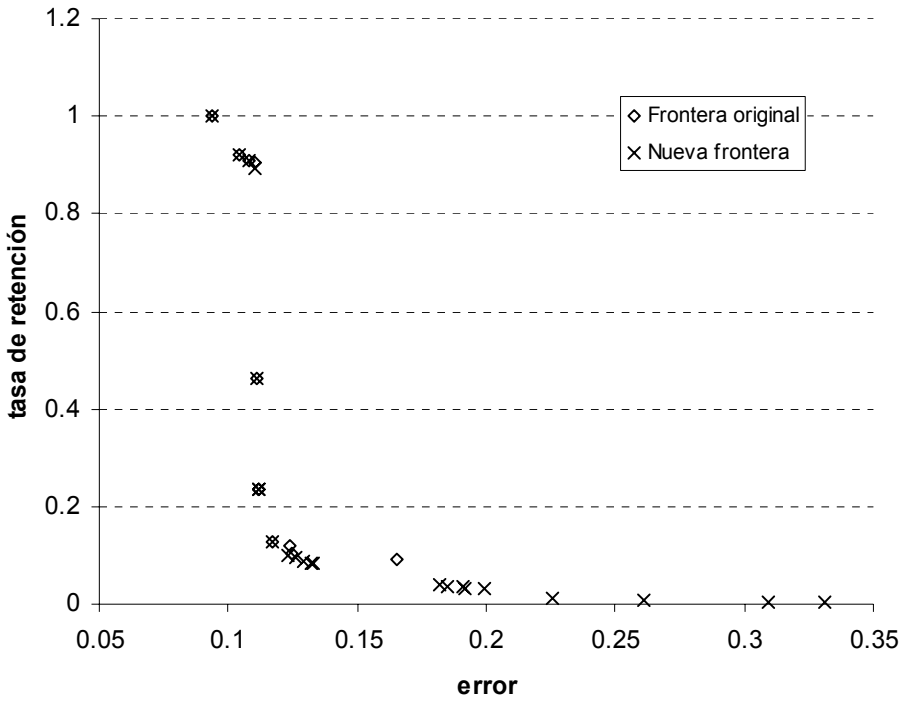


Figura 29. Fronteras de Pareto en base de datos Phoneme (HVDM), antes y después de incluir los nuevos métodos

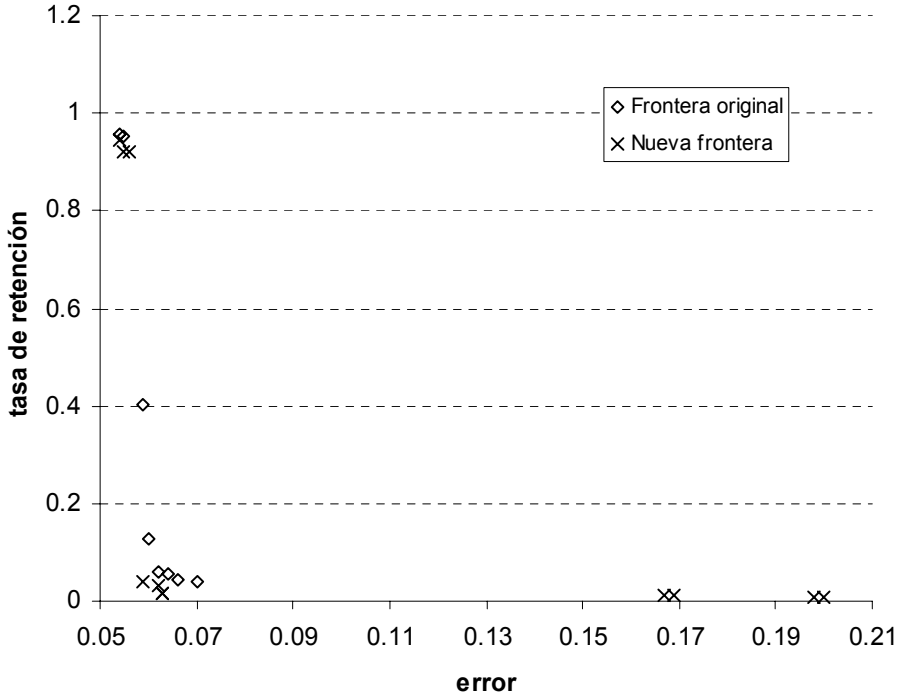


Figura 30. Fronteras de Pareto en base de datos ThyroidDiseaseAnn (HVDM), antes y después de incluir los nuevos métodos

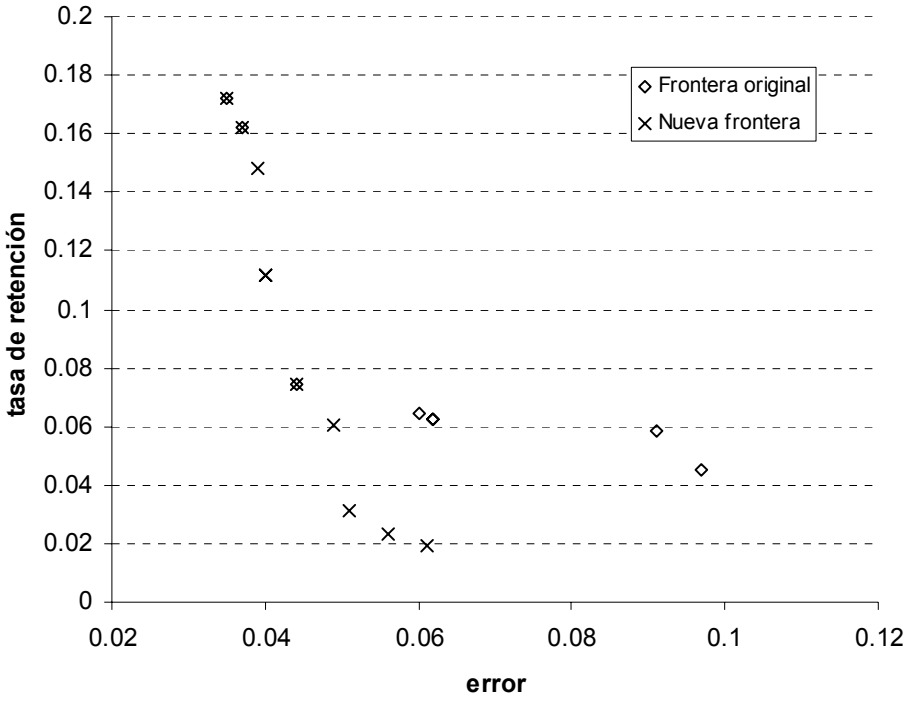


Figura 31. Fronteras de Pareto en base de datos WDBC (HVDM), antes y después de incluir los nuevos métodos

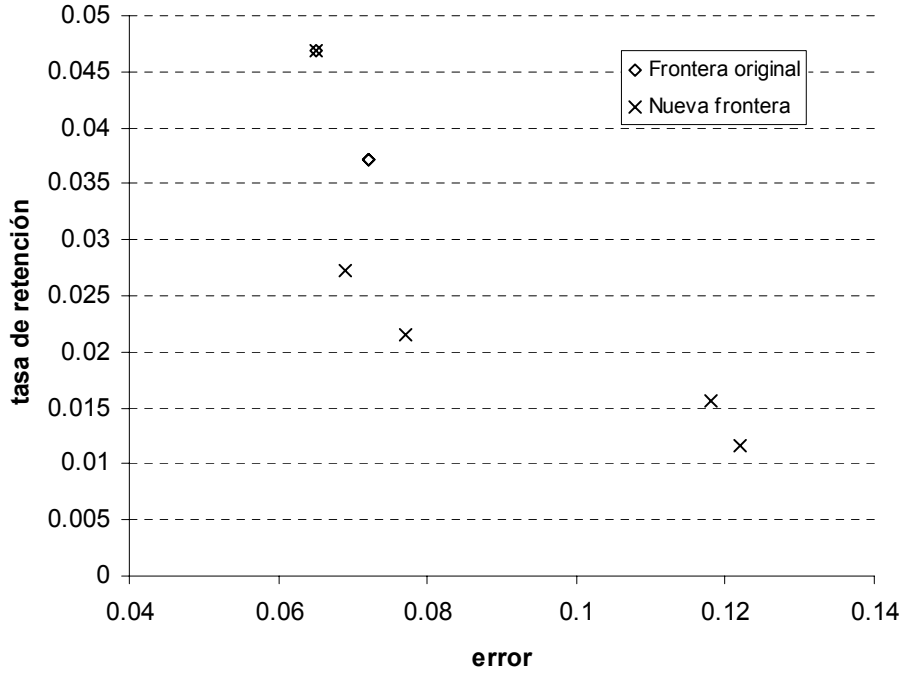


Figura 32. Fronteras de Pareto en base de datos WDBC (Euclidean), antes y después de incluir los nuevos métodos

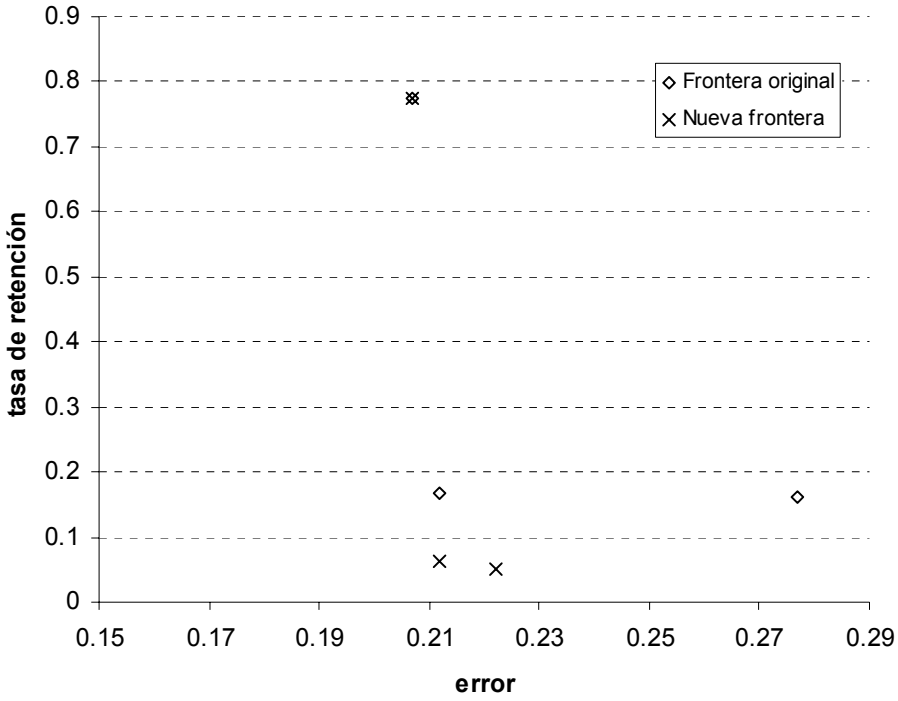


Figura 33. Fronteras de Pareto en base de datos WPBC (HVDM), antes y después de incluir los nuevos métodos

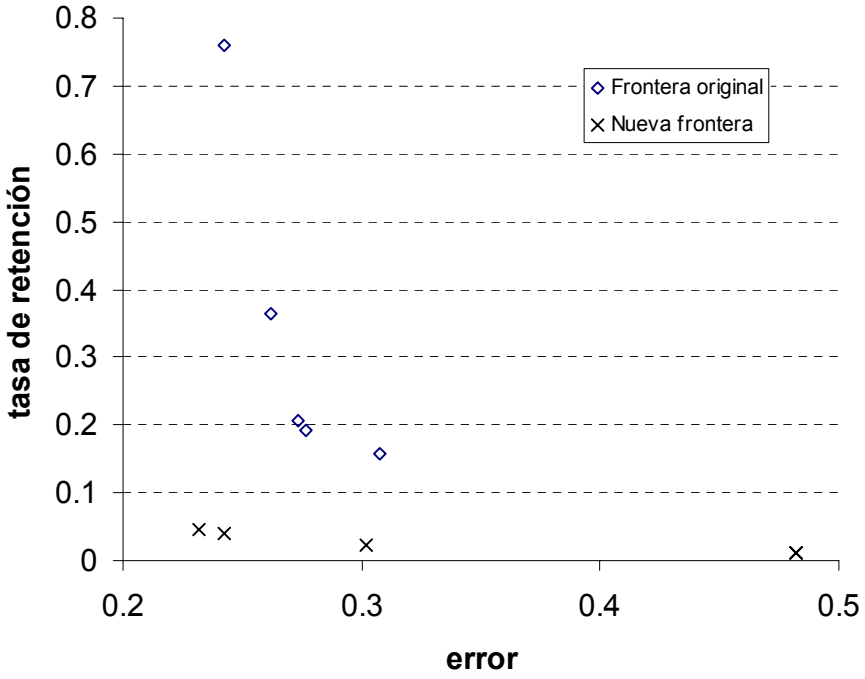


Figura 34. Fronteras de Pareto en base de datos WPBC (Euclidean), antes y después de incluir los nuevos métodos