MÉTODOS PARA MEJORAR LA CALIDAD DE UN CONJUNTO DE DATOS PARA DESCUBRIR CONOCIMIENTO

por

Luis Alberto Daza Portocarrero

Tesis sometida en cumplimiento parcial de los requisitos para optar el grado de

DOCTOR EN FILOSOFÍA
en
Ciencias e Ingeniería de la Información y Computación

UNIVERSIDAD DE PUERTO RICO
Recinto Universitario de Mayagüez

Julio 2007

Aprobado por:

_____	_____
Dr. Edgar Acuña Presidente, Comité Graduado	Fecha
_____	_____
Dr. Julio C. Quintana Miembro, Comité Graduado	Fecha
_____	_____
Dr. Tokuji Saito Miembro, Comité Graduado	Fecha
_____	_____
Dr. Pedro Vásquez-Urbano Miembro, Comité Graduado	Fecha
_____	_____
Dr. Raúl Macchiavelli Representante OEG	Fecha
_____	_____
Dr. Néstor Rodríguez Director del programa en CISE	Fecha

ABSTRACT

Today, data generation is growing exponentially in both directions; instances (rows) and features (columns). This causes that many datasets can not be analyzed without preprocessing. The large size of the dataset to be analyzed may produce serious problems to some data mining algorithms in scalability as well in performance. On the other hand the quality of the data could be inadequate for the knowledge discovery process. For this reason, it is necessary to preprocess the dataset to make it suitable for an efficient performance of the data mining algorithm, and in order to obtain accurate results from it. In this thesis, we introduced new measures to evaluate the quality of a dataset in the context of supervised classification. From these quality measures, we obtain two ways of quantifying the data complexity for a classification problem, specifically, we try to anticipate the behavior of a classification algorithm given a dataset. Our data complexity measures are compared with others already available in the literature, and they give similar performance, but with a lower computational cost. For data cleaning, we propose a new method, which is independent of the classification algorithm. The proposed method detects and eliminates the noise in each class. Our method performs with more efficiency and accuracy than other methods already available in the literature. In the context of dimensionality reduction, we propose two new methods for feature selection. These methods are compared with two well known feature selection methods, the RELIEF and the Sequential Forward Selection (SFS), and similar results are obtained but with a much lower computational costs. Furthermore, we propose a new algorithm, which improves the scalability of the algorithms for instance selection currently in use Finally, we integrate the three processes: data cleaning, reduction of dimensionality, and instance selection, in order to generate a training set, which it will permit an efficient performance of the data mining algorithms, yielding accurate results.

RESUMEN

En la actualidad, existe un crecimiento exponencial en la generación de datos, este crecimiento de los datos se da en ambas dimensiones: filas y columnas, lo que provoca que mucha información no pueda ser procesada y analizada sin un previo procesamiento. El gran tamaño del conjunto de datos a ser analizado puede causar serios problemas a los algoritmos de la minería de datos tanto en su escalabilidad como en su rendimiento. Por otro lado, la calidad de los datos no necesariamente es la adecuada para un proceso de extracción de conocimiento. Por ello se hace necesario que los datos sean preprocesados con la finalidad de adecuarlos a las técnicas existentes y éstas puedan trabajar de manera eficiente y generar resultados precisos. En esta investigación se proponen medidas para evaluar la calidad de los datos en el contexto de la clasificación supervisada. De estas medidas de calidad, se obtienen dos formas de cuantificar la complejidad de un conjunto de datos con respecto al problema de clasificación, específicamente, se trata de anticipar el comportamiento de un algoritmo de clasificación dado un conjunto de datos. Nuestras propuestas para medir la complejidad de los conjuntos de datos, es comparada con otras propuestas, mostrando un rendimiento similar a un menor costo computacional. Para la limpieza de los datos, se propone una metodología que es independiente de los algoritmos de clasificación, y la cual detecta y elimina el ruido en las clases. Nuestro método de reducción de ruido es comparado con otras propuestas, mostrando una mayor eficiencia y precisión. En el contexto de la reducción de la dimensionalidad, se proponen dos métodos eficientes y precisos de selección de variables. El rendimiento de nuestros dos métodos son comparados con dos métodos tradicionales de selección de variables: el RELIEFF y la selección secuencial hacia adelante (SFS) mostrando resultados comparables, pero con menores costos computacionales. También, se propone un nuevo algoritmo que mejora la escalabilidad de los algoritmos de selección de instancias ya existentes. Finalmente, se integra el proceso de limpieza de los datos, selección de variables reducción y selección de instancias, a fin de generar un conjunto de entrenamiento que permita a los algoritmos de la minería de datos trabajar de manera eficiente y que los resultados sean más precisos.

DEDICATORIA

A mi madre por todo su sacrificio, apoyo permanente y por sus sabios consejos y enseñanzas.

A Ana Cristina mi preciosa hija que me dio la fuerza e inspiración final para terminar la investigación.

A mi adorada esposa Mileidys, por todo su apoyo, comprensión y amor.

AGRADECIMIENTOS

Al Dr. Edgar Acuña Fernández, presidente de mi Comité Graduado, por sus sugerencias, orientación y apoyo constante en el desarrollo del presente trabajo de investigación y mis estudios doctorales.

A los miembros de mi Comité graduado, Dr. Julio Quintana, Dr. Tokuji Saito y Dr. Pedro Vásquez-Urbano, por sus sugerencias, correcciones y recomendaciones en el desarrollo y presentación de esta tesis.

A la Oficina de Investigación Naval (ONR) por el apoyo económico parcial recibido a través del Grant N0014-03-0359.

Al Departamento de Defensa por el apoyo económico parcial recibido a través del Grant N0014-06-1-0555.

A todos los que fueron mis profesores del departamento de Matemáticas e Ingeniería del RUM, por sus invalorables enseñanzas.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

Capítulo I
Introducción

1.1. Justificación

La velocidad con que se generan y almacenan datos, es muy superior a la velocidad con que se procesan y analizan (Hernández *et al*., 2004). Se tiene un crecimiento exponencial en la cantidad de datos que son generados por diferentes empresas, gobiernos, instituciones educativas y personas individuales. Existen diversas razones para este incremento, principalmente de índole tecnológico, esto es, el uso intensivo del computador, el incremento en la capacidad de los medios para almacenamiento de datos y sus bajos costos (Fayyad *et al*., 1996). Por otro lado, ha surgido lo que se denomina flujo constante de datos, que se caracteriza en que los datos llegan constantemente sin parar (Frawley *et al*., 1991). De esta forma el incremento en la cantidad de datos crece en forma permanente a un ritmo bastante alto (Li, 2002; Kantardzic, 2001). Esta abundancia de datos ha creado la gran necesidad de metodologías para analizar y explotar la información contenida en esos datos, caracterizados por tener muchas instancias. La principal preocupación que se tiene, es de cómo obtener conocimiento útil de esta avalancha de información.

El crecimiento de los datos se da en ambas dimensiones: tanto en filas (número de instancias) y columnas (número de variables). Existen muchas técnicas de aprendizaje automático, métodos estadísticos, inteligencia artificial y algoritmos especializados para la minería de datos, que tienen serios problemas para poder trabajar eficientemente con grandes cantidades de datos. La enorme cantidad de información afecta la escalabilidad y la precisión. Por ejemplo, en el caso de conjuntos de datos con alta dimensionalidad, la presencia de variables irrelevantes y redundantes degradan el rendimiento de los

algoritmos de la minería de datos (Yang y Pederson, 1997; Xing *et al*., 2001). Por lo tanto, se hace necesario, seleccionar un subconjunto de variables para facilitar y mejorar la labor de la minería de datos. Por otro lado, un gran número de filas hace que el proceso de extraer conocimiento de grandes conjuntos de datos, se convierta en un problema muchas veces intratable con los algoritmos disponibles. La reducción de datos, busca identificar cuáles son los adecuados para realizar el análisis. Esta reducción se aplica con la finalidad de obtener una representación reducida del conjunto de datos original, mucho más pequeña, pero manteniendo la integridad del conjunto de datos completo. Esto es, la minería de datos será más eficiente y se podrán producir los mismos (o casi los mismos) resultados (Han y Kamber, 2006; Zhu y Wu, 2006; Liu y Motoda, 2002).

Ante esta situación se hace necesaria la aplicación de técnicas de reducción de instancias y variables, antes de aplicar alguna técnica de la minería de datos. Debido a dos razones: primero, se tienen problemas para validar e interpretar los resultados, ya que se generan modelos demasiado complejos y muchas veces se pueden tener severos errores de predicción. La segunda razón, es que la carga computacional de construir modelos y validarlos utilizando los algoritmos existentes de la minería de datos es extremadamente alta.

Por muchos años, el problema de clasificación se ha convertido en un importante tópico de investigación en las áreas de aprendizaje automático, reconocimiento de patrones, estadística, y recientemente dentro de las comunidades de bases de datos y minería de datos (Han y Kamber, 2006; Hernández *et al.*, 2004). De esta forma, la clasificación es una tarea fundamental en la minería de datos que será tratada en la presente investigación.
El desarrollo de esta tesis se enfoca en el pre-procesamiento de los conjuntos de datos en problemas de clasificación supervisada. El pre-procesamiento de los datos está compuesto por una serie de procedimientos tendientes a mejorar la calidad del conjunto de datos, corrigiendo las inconsistencias, completando valores perdidos, detectando y reduciendo el ruido, identificando los valores anómalos, seleccionando variables e instancias, etc. (Kim *et al*., 2003; Han y Kamber, 2006; Hernández *et al.*, 2004). Nosotros específicamente nos

enfocamos en las tareas de limpieza de datos (se identifica y remueve el ruido en las clases), selección de variables e instancias. En resumen, el presente estudio está abocado a la búsqueda de una reducción en el volumen de datos original, seleccionando los datos más relevantes que mejoren la calidad del conjunto de datos y que el conocimiento descubierto sea más preciso.

Los métodos propuestos y analizados, se fundamentan en el cálculo de la complejidad de un problema de clasificación. Varios estudios han mostrado que el comportamiento empírico de los clasificadores está fuertemente relacionado a los datos disponibles. *A priori*, es muy difícil determinar que clasificador tendrá un mejor rendimiento, dado un problema específico. Generalmente, se han realizado estudios experimentales con varios clasificadores, y se han reportado los errores de clasificación en un número limitado de problemas, sin analizarse el porqué de las diferencias en el rendimiento de los clasificadores para cada problema de clasificación. En este contexto, resulta razonable determinar la complejidad del conjunto de los datos antes de usar los algoritmos de clasificación existentes. La estimación de la complejidad, nos puede permitir predecir el comportamiento de los clasificadores individuales o combinados y realizar selección de variables y/o instancias para un problema de clasificación en particular basadas en estas mismas medidas de complejidad.

En esta investigación se proponen algoritmos eficientes y precisos para identificar el ruido en las clases. La limpieza de instancias ruidosas en el conjunto de entrenamiento, contribuye a mejorar el rendimiento de los clasificadores y de simplificar la tarea de los mismos al reducir el tamaño del conjunto de datos y suavizar la frontera de decisión, para finalmente producir modelos más precisos. Además, se desarrollan dos métodos de selección de variables para reducir la dimensionalidad del conjunto de entrenamiento. El objetivo de los métodos propuestos de selección de variables es el de eliminar eficientemente los atributos irrelevantes y redundantes, con la finalidad de mejorar la precisión y eficiencia de los algoritmos de clasificación. Finalmente se trata el problema de la selección de instancias, que consiste en obtener un subconjunto del conjunto de

entrenamiento de forma que se garantice que las instancias relevantes y representativas sean seleccionadas en una muestra de tamaño óptimo que nos permita obtener un rendimiento eficiente del algoritmo de la minería de datos con una pérdida mínima en su poder de predicción en los problemas de clasificación supervisada. También se propone un esquema que combina en forma secuencial, la detección de ruido, la selección de variables y la reducción del número de instancias, para mejorar la calidad del conjunto de entrenamiento, con el que los algoritmos de la minería de datos puedan trabajar de manera eficiente y generar modelos menos complejos y más precisos.

1.2. Objetivos

Objetivo General:

Desarrollar medidas de complejidad de los conjuntos de datos en problemas de clasificación supervisada e implementar técnicas eficientes de detección de ruido, selección de variables e instancias para optimizar el rendimiento de los algoritmos de la minería de datos.

Objetivos específicos:

- Construir medidas eficientes de la complejidad de los conjuntos de datos, que permitan describir el comportamiento de los algoritmos en problemas de clasificación supervisada.

- Desarrollar e implementar un algoritmo para identificar y eliminar el ruido en el conjunto de entrenamiento con la finalidad de mejorar la precisión de los clasificadores.

- Elaborar métodos eficientes de selección de variables basados en la complejidad del problema de clasificación.

- Mejorar la escalabilidad de los algoritmos de selección de instancias, que aumente su capacidad de reducir el conjunto de entrenamiento con un menor costo computacional.

- Combinar la detección y eliminación de ruido, con la selección de variables e instancias con la finalidad de mejorar el rendimiento de los clasificadores.

1.3. Organización de la tesis

La presente tesis está organizada en nueve capítulos. En el segundo capítulo se presentan los antecedentes del trabajo de investigación, revisando las definiciones del proceso de extracción de conocimiento de bases de datos, además de ubicar dentro de este proceso el desarrollo de esta investigación.

En el tercer capítulo se discute la complejidad de los problemas de clasificación supervisada. Se proponen medidas de la complejidad que son comparadas con otras medidas existentes en la literatura.

En el cuarto capítulo se proponen dos métodos de selección de variables y se realizan comparaciones con otros métodos de selección utilizando conjuntos de datos reales.

El quinto capítulo está dedicado a tratar el problema de ruido en las clases en problemas de clasificación supervisada. Se describe el problema y sus efectos sobre la precisión del clasificador y se revisan los antecedentes de este problema. Finalmente se propone y se compara un método de detección y eliminación de ruido en las clases.

En el sexto capítulo se trata el problema de la selección de instancias. Se revisan los algoritmos existentes y se propone un algoritmo que mejora la escalabilidad de los algoritmos de selección de instancias. Se realizan comparaciones experimentales entre el algoritmo propuesto y el método clásico.

En el capítulo séptimo, se presentan resultados experimentales correspondientes a los efectos de eliminar el ruido en las clases y realizar en forma combinada la selección de variables e instancias.

El capítulo octavo, corresponde a la presentación de las conclusiones finales y el trabajo futuro.

Finalmente en el capítulo noveno se hace una reflexión sobre los aspectos éticos de la investigación.

Capítulo II

Conceptos Previos

2.1. Descubrimiento de conocimiento en bases de datos

El reto al cual nos enfrentamos en la actualidad es el de trabajar con grandes cantidades de datos, que crecen a un ritmo exponencial, de tal forma, que se hace imposible procesar toda la información que se genera. La respuesta a este reto se da a través del uso de las técnicas del descubrimiento de conocimientos de bases de datos (KDD, de sus siglas en inglés). Según Frawley *et al*., (1991); Fayyad *et al*. (1996) y Liu y Motoda, (1998), la definición clásica de KDD es:

> El descubrimiento de conocimiento en bases de datos es el proceso de extracción no trivial para identificar patrones que sean válidos, novedosos, potencialmente útiles y entendibles, a partir de los datos.

Al KDD a veces se le conoce como minería de datos (DM, de sus siglas en inglés). Sin embargo, esencialmente el proceso de minería de datos envuelve la aplicación de un algoritmo para extraer patrones de datos y al KDD como el proceso completo (pre-procesamiento, minería de datos, post-procesamiento, ver figura 2.1). De esta forma el proceso KDD está compuesto por una serie de etapas en la que DM es la encargada de extraer (identificar) modelos a partir de las bases de datos disponibles (Hernández *et al*., 2004).

Figura 2.1. El proceso KDD

2.2. Proceso de descubrimiento de conocimiento

De acuerdo con Hernández *et al.*, (2004), el proceso de descubrimiento de conocimiento en bases de datos involucra varios pasos:

1. Entendimiento del dominio de aplicación, el conocimiento relevante a usar y las metas del usuario.
2. Seleccionar un conjunto de datos y enfocar la búsqueda en subconjuntos de variables y/o muestras de datos en dónde realizar el proceso de descubrimiento.
3. Limpieza y pre-procesamiento de los datos, diseñando una estrategia adecuada para manejar el ruido, valores incompletos, datos redundantes, datos inconsistentes, etc.
4. Reducción de datos y proyecciones para reducir el número de variables a considerar
5. Selección de la tarea de descubrimiento que se va a realizar, por ejemplo, clasificación, agrupamiento, regresión, etc.
6. Selección del o de los algoritmos a utilizar.
7. Llevar a cabo la etapa de la minería de datos.
8. Interpretar los resultados.
9. Incorporar el conocimiento descubierto al sistema.

2.3. Minería de datos (DM)

La DM integra una serie de áreas tales como, el aprendizaje automático, métodos estadísticos, inteligencia artificial y métodos especializados para la minería de datos, con el propósito de identificar y extraer conocimiento a partir de las bases de datos que se orienta hacia la toma de decisiones (Han y Kamber, 2006; Hernández *et al*., 2004).

2.3.1. Tareas de la minería de datos

Según Han y Kamber, 2006 y Hernández *et al*., 2004; las principales tareas de la minería de datos son:

- **Análisis de dependencia**: El valor de un elemento puede usarse para predecir el valor de otro. La dependencia puede ser probabilística y puede definir una red de dependencias o puede ser funcional. El análisis de dependencias se ha orientado, durante los últimos años, al descubrimiento de redes Bayesianas o causales en donde la dependencia se da a nivel estructural (dependencias e independencias entre variables) y cuantitativa (fuerza de las dependencias).
- **Identificación de clases o grupos**: Identifica un conjunto finito de categorías o *clusters* que describen a los datos. Las clases pueden ser relevantes entre sí o pueden servir como entradas a otros sistemas de aprendizaje.
- **Descripción de conceptos**: Resume cierto patrón característico. La descripción puede ser característica (qué instancias y valores son comunes entre clases) o discriminatoria (cómo difieren las clases). La mayoría de los sistemas de aprendizaje encuentran descripción de conceptos y están enfocados a la clasificación. Otra técnica relacionada es el análisis de regresión.
- **Detección de desviaciones**: Detectar los cambios más significativos en los datos con respecto a valores pasados o normales. Sirve para filtrar grandes volúmenes de datos que son menos probables de ser interesantes. El principal problema está en determinar cuándo una desviación es significativa para ser de interés.
- **Detección de *outliers***: Otra de las tareas fundamentales en la minería de datos, es la detección de instancias u observaciones anómalas. La presencia de las instancias anómalas o valores extremos, podrían afectar las tareas que se realizan en la minería de datos. No existen muchos estudios que analicen estos efectos, un reciente estudio realizado por Acuña y Rodríguez (2004), encontraron que los clasificadores se ven afectados por la presencia de valores anormales, pero el efecto es relativo y su impacto depende del clasificador y del conjunto de datos. Por otro

lado, la detección de valores anómalos tiene muchas aplicaciones directas, entre las que tenemos: la detección de fraude y la detección de ataques a redes de computadoras. Existen diferentes métodos para detectar *outliers:* se disponen de técnicas estadísticas, basadas en *clustering* y técnicas procedentes de la minería de datos.

2.4. Minería de datos y clasificación

El problema de clasificación se ha constituido por décadas en un importante tópico de investigación en las áreas de aprendizaje automático, reconocimiento de patrones y estadística, y recientemente en las comunidades de bases de datos y minería de datos. De esta forma, la clasificación es una tarea fundamental en la minería de datos. Este es un problema de análisis multivariado conocido como clasificación y tiene dos variantes, la clasificación supervisada y la clasificación no supervisada.

- **Clasificación supervisada**: Es una técnica multivariada relacionada con la asignación de nuevos objetos (observaciones, casos, individuos, instancias) a grupos previamente definidos, haciendo uso de la información proporcionada por un conjunto de variables. Esto significa que existe una relación entre una variable categórica, que determinan los grupos y un conjunto de variables interrelacionadas, denominadas predictoras.
- **Clasificación no supervisada**: En este caso, el problema consiste en descubrir grupos de observaciones que puedan formar varios conglomerados homogéneos. Para esto, se tiene un conjunto de datos en el que no hay una variable categórica que defina un grupo de pertenencia.

2.5. Pre-procesamiento de los datos

Las bases de datos del mundo real son altamente susceptibles al ruido, valores perdidos, datos inconsistentes, datos redundantes, etc., debido fundamentalmente a su gran tamaño y de donde se obtienen, principalmente de múltiples fuentes heterogéneas. Si se dispone de

datos de baja calidad, entonces es lógico esperar que los resultados obtenidos mediante alguna técnica de minería de datos sea también de baja calidad. El pre-procesamiento de los datos busca mejorar la calidad de los datos, y consecuentemente, los resultados de la minería de datos. Adicionalmente, el pre-procesamiento de los datos facilita el trabajo de la minería de datos mejorando su eficiencia, ya que se tendrán conjuntos de datos de menor tamaño al original (Han y Kamber, 2006).

La figura 2.2, muestra las estrategias que se pueden utilizar para mejorar la calidad de los conjuntos de datos disponibles (Cano *et al.*, 2003).

- **Limpieza de datos:** Está compuesta por una serie de procedimientos tendientes a mejorar la calidad del conjunto de datos, corrigiendo las inconsistencias, llenando los valores perdidos, detectando y reduciendo el ruido, identificando los valores anómalos, etc. (Kim *et al*., 2003).
- **Reducción de datos**: Busca identificar cuáles son los datos adecuados para realizar el análisis. Se aplica con la finalidad de obtener una representación reducida del conjunto de datos original, mucho más pequeña, pero manteniendo la integridad del conjunto de datos completo. Esto es, la minería de datos será más eficiente y se podrán producir los mismos (o casi los mismos) resultados analíticos. El presente estudio está abocado en parte a la búsqueda de una reducción en el volumen de datos original, seleccionando los datos más relevantes que van a ser usados por la minería de datos (Han y Kamber, 2006; Liu y Motoda , 2002).

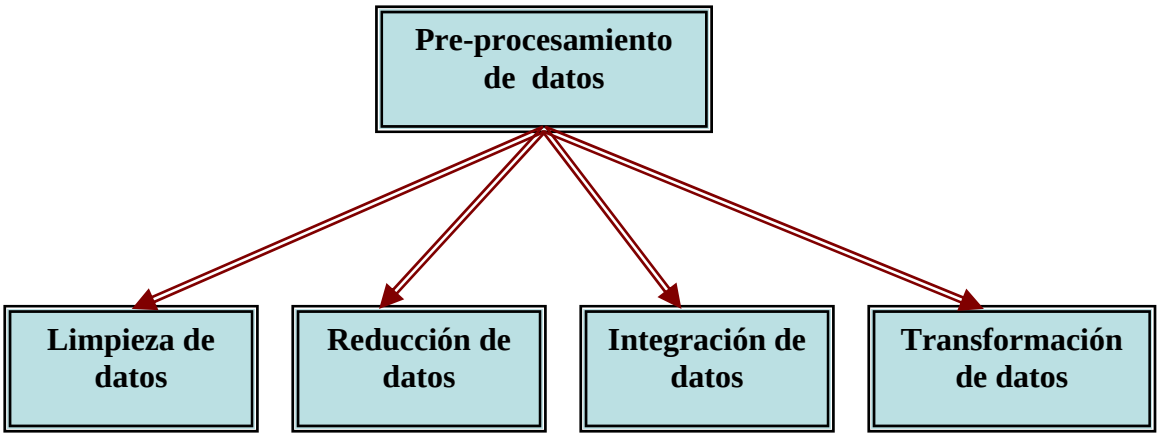

Figura 2.2. Estrategias del pre-procesamiento de datos

- **Integración de datos**: Consiste en combinar datos de múltiples fuentes y almacenarlos en una base de datos coherentemente estructurada. La combinación de datos también incluye la agregación, que consiste en realizar operaciones para obtener nuevos valores a partir de la unión de la información de varios registros o tablas (Han y Kamber, 2006).

- **Transformación de datos**: Los datos son transformados o consolidados en formas apropiadas para que la minería de datos pueda realizar su trabajo. La transformación de datos incluye procedimientos como la normalización, donde cada variable cambia de escala, la construcción de nuevas variables, etc.

2.6. Técnicas de reducción de datos

La reducción de datos se puede realizar usando diferentes técnicas. Entre las técnicas utilizadas para la reducción de datos se tienen al agrupamiento de datos, la discretización, la compresión o compactación de datos, la selección de variables y la selección de instancias.

- **Agrupamiento de datos**: Dado un conjunto de datos en la que no se dispone de una columna de clases o grupos, las técnicas de agrupamiento de datos se utilizan en la clasificación no supervisada para separar las instancias en grupos. Los grupos obtenidos deben reflejar las diferencias entre grupos y las similitudes y relaciones existentes entre las instancias que pertenecen a un mismo grupo.
- **Discretización de los datos**: Esta técnica se usa para reducir el número de valores para una variable continua, mediante la división del rango de valores de la variable en intervalos. Los intervalos generados son etiquetados y se usan para reemplazar los valores actuales de los datos. Algunos métodos de la minería de datos requieren que los conjuntos de datos posean valores discretos para sus variables, en este contexto, si se dispone de un conjunto con variables continuas, se usa la discretización con la finalidad de reemplazar los valores continuos y hacer factible la aplicación del método a dicho conjunto.
- **Compactación de datos**: En la compactación de datos se aplican codificaciones o transformaciones a los datos con el objetivo de obtener una representación reducida o comprimida de los datos originales. Si los datos originales pueden ser reconstruidos partiendo de los datos comprimidos sin pérdida de información, la técnica usada se denomina *lossless.* Si por el contrario, sólo podemos reconstruir una aproximación a los datos originales, entonces la técnica de compresión es llamada *lossy*. Dos de las técnicas más populares de la compactación de datos *lossy* son: las transformaciones *wavelet* y el análisis de componentes principales (Han y Kamber, 2006).
- **Selección de variables**: Los conjuntos de datos a ser analizados, muchas veces tienen un gran número de variables, muchas de las cuales pueden ser irrelevantes o redundantes para la tarea de la minería de datos. Un buen conjunto de variables debe destacar lo mejor posible la similitud entre objetos de la misma clase y la diferencia entre objetos de clases diferentes. Una solución indirecta a esta cuestión consiste en utilizar todas las variables que pudieran aportar información relevante al problema que se trata de resolver, pensando que a falta de conocer la calidad, la cantidad puede ayudar. No obstante, una cantidad importante de atributos aumenta

la complejidad del clasificador. Colateralmente, la presencia de variables redundantes, y posiblemente irrelevantes, pueden afectar negativamente a la confiabilidad del clasificador. Por lo tanto, se hace necesario seleccionar un subconjunto mínimo de variables, de forma tal que la distribución de probabilidad resultante de las clases de datos, esté lo más cercana posible a la distribución original con todas las variables. Un beneficio adicional de la selección de variables es que el número de variables que van a aparecer en los patrones resultantes se va a reducir, y esto significa que los patrones serán más fáciles de entender e interpretar.

- **Selección de instancias**: Consiste en identificar y seleccionar las instancias más relevantes de un conjunto. La finalidad es que la tarea de minería de datos pueda realizarse de manera eficiente, preservando las cualidades del conjunto completo. En el capítulo 6 se detalla en qué consiste la selección de instancias y los procedimientos existentes para poder llevarlo a cabo.

La selección de instancias en el contexto de clasificación ha sido estudiada por varios investigadores. La idea fundamental es que los métodos de selección de instancias deben ser capaces de cumplir con sus tres funciones (limpiar, habilitar y enfocar), además de seleccionar las instancias relevantes y que el tamaño de muestra se determine dinámicamente. La manera más simple e intuitiva de llevar a cabo la selección de instancias es la de realizar un muestreo aleatorio, que se describe en la siguiente sección.

2.7. Muestreo

Según Cochran (1977), el muestro es un procedimiento mediante el cual se obtiene un subconjunto de instancias de tamaño n, de una población de tamaño N, mediante un proceso aleatorio en el que cada uno de los subconjuntos de instancias de tamaño n recibe una probabilidad apropiada de ser seleccionado.

El proceso de muestreo se puede realizar de dos maneras. La primera forma, se utiliza en encuestas o experimentos aleatorios y consiste en tomar una muestra de una población más grande, desconocida. En este caso, el proceso de muestreo forma parte del proceso de la recolección de datos. La minería de datos no está interesada en este tipo de muestreo. La

segunda forma, constituye el procedimiento más utilizado en el contexto de la minería de datos. Los datos completos ya están dados de antemano y representan una población sumamente grande y el análisis de los datos sólo se basa en un subconjunto de instancias (Kantardzic, 2001; Toivonen, 1996).

Las técnicas de muestreo clásico o muestreo por lotes tienen dos ventajas que es importante resaltar. La primera ventaja, es que el costo computacional es bastante bajo, los tiempos de computación son del orden $O(n)$, esto es, tiene un comportamiento lineal en el tamaño de la muestra seleccionada. La segunda ventaja es que constituyen un modelo de selección de instancias independiente, debido a que no utilizan ningún algoritmo de la minería de datos para la extracción de las instancias.

2.7.1. Tipos de Muestreo

En esta sección se presenta una breve descripción de algunas técnicas de muestreo clásico más utilizadas.

- **Muestreo sistemático**: Es la técnica de muestreo más simple. Consiste en seleccionar la muestra eligiendo las instancias cada cierto intervalo en la secuencia ordenada de instancias. Por ejemplo, si se desean seleccionar el 50% de un conjunto de datos, el proceso consiste en seleccionar una instancia de cada dos en la base de datos.
- **Muestreo Simple Aleatorio:** El muestreo simple aleatorio consiste en seleccionar al azar una muestra de tamaño n, de una población de tamaño N. En este tipo de muestreo todas las instancias tienen la misma probabilidad de ser seleccionados. La selección de la muestra puede realizarse a través de cualquier mecanismo probabilístico en el que todos los elementos tengan las mismas opciones de salir. Este método tiene dos variantes: muestreo al azar sin reemplazo y muestreo al azar con reemplazo.
- **Muestreo Aleatorio Estratificado:** El conjunto de datos (población) de tamaño N, es dividido en k subconjuntos homogéneos bajo algún criterio o característica, de

tamaños $N_1, N_2, \cdots, N_k$, respectivamente. Estos subconjuntos son disjuntos y se forman utilizando un criterio o una característica en común. Se debe cumplir que $\sum N_i = N$. Los subconjuntos se denominan estratos. Luego de que los estratos han sido determinados, se toma una muestra aleatoria simple de cada estrato, la selección en cada estrato se realiza en forma independiente. El tamaño de muestra para cada uno de los estratos se denota por $n_1, n_2, \cdots, n_k$, respectivamente.

- **Muestreo Aleatorio Ponderado.** A diferencia del muestreo simple al azar, las instancias son elegidas con cierta probabilidad (que difiere de una instancia a otra), a esta probabilidad se le denomina "peso". La asignación del peso a cada observación depende de la aplicación, en clasificación normalmente se asignan los pesos utilizando una función de densidad.
- **Muestreo Adaptable.** El principio del muestreo adaptable consiste en evaluar y tomar una decisión sobre la base de la muestra que se ha elegido en un instante dado, esto es, el muestreo adaptable consiste en muestrear seleccionando las instancias que deben ser incluidas en la muestra, dependiendo del resultado que se obtenga con la muestra seleccionada en el paso previo. El propósito primario del muestreo adaptable es aprovecharse de las características de los datos para obtener estimaciones más precisas.

Sus variantes pueden encontrarse en el muestreo secuencial o muestreo en línea y el muestreo progresivo. Estos casos particulares del muestreo adaptable utilizan el valor del parámetro que se va a estimar para obtener una muestra representativa, en el caso de clasificación será el error de clasificación. En muchos artículos se considera propiamente adaptables cuando el algoritmo no utiliza el valor de lo que se va a estimar como una cota al tamaño de la muestra, esto es, el criterio de parada es independiente de lo que se quiere estimar. En términos generales el muestreo adaptable es un problema de optimización y debe existir una función objetivo que debe minimizarse o maximizarse.

2.7.2. Tamaño de muestra

De acuerdo con Kantardzic (2001), hay varias estrategias para obtener el tamaño de muestra de un subconjunto representativo de instancias de un conjunto de datos. El tamaño de muestra adecuado se determina tomando en cuenta los costos de computación, requerimientos de memoria, exactitud del estimador, y otras características del algoritmo y los datos. Estadísticamente, se determina el tamaño de un subconjunto de datos de tal forma que el estimador para el conjunto de datos completo difiera del estimador usando el subconjunto en a lo más un margen de error absoluto de ε. Para determinar este tamaño de muestra se plantea la siguiente desigualdad, en términos del nivel de tolerancia al error (o nivel de precisión):

$$P\left(\left| \hat{\theta}_n - \theta \right| \geq \varepsilon \right) \leq \delta \tag{2.1}$$

Esta misma ecuación, expresada en términos de la precisión relativa, puede expresarse como:

$$P\left(\left| \frac{\hat{\theta}_n - \theta}{\theta} \right| \geq \varepsilon \right) \leq \delta \tag{2.2}$$

Resolviendo una de las desigualdades (2.1) ó (2.2), para *n*, se obtiene el tamaño del subconjunto de muestra *n*, para un valor dado de ε (margen de error o nivel de precisión) y δ (donde $1-\delta$ es el nivel de confianza). $\hat{\theta}$ simboliza un estimador obtenido con un subconjunto de instancias y generalmente es una función del tamaño del subconjunto *n*, mientras que θ es el parámetro, esto es, el valor verdadero obtenido del conjunto completo de datos o la población. Sin embargo, θ también es usualmente desconocido. En este caso, una forma práctica para determinar el tamaño requerido para el subconjunto de los datos consiste en proceder mediante los siguientes dos pasos:

Paso 1. En el primer paso se selecciona una muestra preliminar pequeña de tamaño *m*.

Paso 2. Con las instancias seleccionadas en el paso 1, se estima el parámetro θ, este estimado se reemplaza en la desigualdad y se resuelve para *n*. Si $n \geq m$, se obtienen $n - m$ instancias adicionales para obtener la muestra final. Si $n \leq m$ entonces no se adicionan instancias adicionales y las instancias seleccionadas en la muestra preliminar se usan como la muestra final.

2.7.3. Cota aditiva de Chernoff ó Cota de Hoeffding

Otra forma de determinar el tamaño de muestra apropiado dado un nivel de precisión ε y un nivel de confianza $1-\delta$, es la cota aditiva Chernoff o los límites de Hoeffding. Estos límites se usan comúnmente en las investigaciones de reconocimiento de patrones o clasificación (Domingos y Hulten, 2002; Kivinen y Mannila, 1994). Por ejemplo, sea r una variable aleatoria que representa la diferencia entre el valor del parámetro y su estimación empírica basada en una muestra de tamaño finito. R representa el rango de r. Supongamos que tenemos n observaciones independientes de esta variable y que se calcula su media $\bar{r}$. La cota de Hoeffding que con un nivel de confianza de $1-\delta$ e independientemente de la verdadera distribución de r, la verdadera media de la variable difiere en ε de $\bar{r}$, donde:

$$\varepsilon = \sqrt{\frac{R^2 \ln\left(2/\delta\right)}{2n}} \tag{2.3}$$

De aquí podemos deducir que si queremos estimar únicamente la media de r, de forma que difiera de $\bar{r}$ en ε, aceptando una probabilidad de fallar igual a δ, se requiere un tamaño de muestra de:

$$n = \frac{1}{2}\left(R/\varepsilon\right)^2 \ln\left(2/\delta\right) \tag{2.4}$$

Una de las principales limitaciones en el uso de estas alternativas para determinar el tamaño de muestra, es que son muy conservadoras, lo que significa que el tamaño de la muestra resultante es muy grande para obtener una exactitud razonablemente buena y confiable. La razón para que esto ocurra, se debe a que el tamaño de muestra es calculado *a priori*, y por lo tanto debe de ser lo suficientemente grande y preciso para trabajar en todas las situaciones que se puedan encontrar (peor de los casos).
Para solucionar el problema de determinar el tamaño de muestra, existe la propuesta de muestrear en forma secuencial en vez de hacerlo por lotes.

2.7.4. Muestreo adaptable en clasificación supervisada

El muestreo clásico ha mostrado que no es adecuado para reducir el tamaño de grandes conjuntos en el campo de la DM. Algunos autores (ver por ejemplo, Wang *et al.*, 1998), indican que esta solución no es recomendable debido a la dificultad de determinar el tamaño apropiado de la muestra a seleccionar. La alternativa es usar el muestro adaptable.

El muestreo adaptable, usa el criterio probablemente más próximo (PCE, de sus siglas en inglés). La idea consiste en seleccionar dinámicamente las instancias e ir evaluando su rendimiento de acuerdo con el criterio PCE. En contraste, el muestreo clásico selecciona un conjunto de instancias y las evalúa una sola vez. El criterio PCE queda expresado en la siguiente fórmula:

$$P\left(\left|Acc(N) - Acc(n)\right| > \in\right) \leq \delta\,, \tag{2.5}$$

donde, $Acc(N)$ es el porcentaje de aciertos con la muestra completa utilizando el algoritmo de minería de datos, $Acc(n)$ es el porcentaje de aciertos con la muestra seleccionada, $\in$ es un parámetro de precisión y el parámetro δ está relacionado con el nivel de confianza.

2.7.4.1. Muestreo secuencial adaptable

El muestreo secuencial se utiliza para determinar adaptablemente el tamaño de muestra. Hace uso de las fórmulas que se utilizan para calcular el tamaño de muestra en el muestreo por lotes, pero se actualiza en forma secuencial cada vez que se van a incluir nuevas observaciones a la muestra. La idea se basa en que uno puede determinar los parámetros iniciales sobre la base del conocimiento de los datos y luego, estimar los parámetros que se requieren en los límites estadísticos o las cotas para el tamaño de muestra. Trabajos en este sentido son los desarrollados por Gavaldá y Watanabe (2000), Domingo *et al*. (2002).

Watanabe (2000) utiliza el muestreo secuencial adaptable para resolver un problema bastante simple. El problema consiste en la estimación de la proporción de instancias con una característica determinada $B(x)$, en un conjunto de datos. Primero formula un algoritmo en el que se fija el tamaño de muestra n y se contabiliza el número de instancias que tienen la característica B. Esta formulación está basada en el muestreo por lotes. Luego, plantea un algoritmo que actualiza de manera secuencial, tanto el valor de la estimación como el del tamaño de muestra. Este estudio, es bastante simple y limitado al contexto de la estimación de proporciones en una población y lo que busca es ilustrar la aplicación del muestreo secuencial.

2.7.4.2. Muestreo adaptable no secuencial

Los trabajos que se han mostrado sobre el muestreo secuencial, utilizan un clasificador para determinar dinámicamente el tamaño de muestra adecuado. También se utiliza un clasificador como medio para evaluar la calidad de la muestra seleccionada, y las instancias que deben de incluirse en la muestra. Naturalmente este enfoque tiende a sesgar los resultados ya que las instancias seleccionadas van a depender del clasificador, por lo tanto se obtienen soluciones particulares.

Li (2002), propone una interesante propuesta de un algoritmo adaptable no secuencial. Su propuesta utiliza el algoritmo de búsqueda discreta, denominado Búsqueda Tabú, para evaluar las observaciones que deben estar en la muestra, minimizando una función objetivo basado en un criterio simplificado que mide la bondad de ajuste de la una muestra y la población de donde proviene. Esta medida está dada por,

$$\chi^2 = \sum \frac{\left(n_i - m_i\right)^2}{m_i} \tag{2.6}$$

donde n_i es la frecuencia observada; m_i es la frecuencia esperada de la distribución; y el subíndice *i* representan todas las posibles combinaciones de las categorías (p.e. todas las celdas en las tablas de contingencia). La estadística χ^2 sigue la distribución asintótica Chi-cuadrado. El problema de calcular esta estadística χ^2, es que es computacionalmente intratable cuando se tiene un conjunto grande de datos con más de cuatro categorías por variable.

Li realiza una simplificación de la estadística χ^2 basada en las distribuciones marginales. La estadística simplificada de la Chi-cuadrado χ_s^2, está dado por,

$$\chi_s^2 = \sum_{j=1}^{J} \sum_{k=1}^{K_j} \frac{\left(n_{jk} - pN_{jk}\right)^2}{pN_{jk}} \tag{2.7}$$

Lo relevante de esta propuesta es que no utiliza ningún clasificador para evaluar las instancias que deben estar en la muestra, si no más bien, la medida de ajuste de la muestra al conjunto de datos completo. Sin embargo, esta propuesta posee varias limitaciones, la primera es que solo es aplicable a un conjunto de datos que tengan atributos discretos o nominales. La segunda limitación es que no considera la

interrelación entre las variables debido a problemas computacionales. Una tercera limitación es que el tamaño de muestra se determina *a priori*. Este método de muestreo adaptable no secuencial es totalmente heurístico, por lo que requiere de un estudio empírico profundo para evaluar su rendimiento, y proponer mejoras. Los resultados experimentales proporcionados por Li son limitados y parciales debido a que no proporciona en su reporte la variabilidad de las estimaciones que obtiene con las muestras seleccionadas. Por otro lado, podría haber estudiado la relación que existe entre el valor de la función objetivo y el error de clasificación. La otra limitación es que puede ser extremadamente lento o no llegar a obtener la muestra óptima con cierto tipo de conjunto de datos, por ejemplo datos con variables que tengan muchas categorías y muchas clases.

2.8. Estimación de la tasa de mala clasificación o error de clasificación

Al aplicar una regla de clasificación es importante saber si el clasificador está realizando su trabajo de manera adecuada, esto es, un objeto que pertenece al i-ésimo grupo es asignado a dicho grupo y no a otro. Este proceso de cálculo, involucra la existencia de dos conjuntos de datos: el conjunto de entrenamiento o aprendizaje y el conjunto de prueba.

- **Conjunto de aprendizaje o entrenamiento**: En la práctica, la mayoría de las veces, las densidades condicionales a los grupos son desconocidas parcial o totalmente, por lo que requieren ser estimadas. Un supuesto básico en el análisis discriminante para estimar las densidades condicionales es que existen objetos con origen conocido para los cuales se tienen las p mediciones del vector de características o variables $X \in R^p$. Sean $\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_n$ cada una de las mediciones observadas en las características o variables y $\underline{z}_1, \underline{z}_2, \cdots, \underline{z}_n$ los correspondientes vectores que indican la pertenencia del objeto a un grupo. El conjunto de datos en la matriz *E*, definida por $E = \{(\underline{x}_i, \underline{z}_i), \quad i = 1, \ldots, n\}$, se denomina muestra de entrenamiento o de aprendizaje.

- **Conjunto de prueba**: Es la muestra que se usa para validar el clasificador. Se asume que se ha extraído independientemente de la misma población de donde se extrajo la muestra de entrenamiento.

Existen varias formas de medir el poder discriminante del clasificador (Hand, 1997), pero la más utilizada es la tasa de mala clasificación o tasa de error del clasificador. Generalmente, es muy difícil obtener una expresión analítica para la tasa de error, por lo tanto debe ser estimada con la información muestral disponible.

La tasa de error verdadero o tasa de error actual de un clasificador, e_t, es la probabilidad de que la regla de clasificación (o simplemente el clasificador), $C(\underline{x})$, clasifique mal un objeto proveniente de una muestra obtenida posteriormente a la muestra de entrenamiento. Esta es la tasa de error obtenida sobre la base de un número infinito de muestras de prueba provenientes de la misma distribución que la muestra de entrenamiento. De esta manera y según lo mencionado en las secciones anteriores, la tasa de error óptima de Bayes, es el mínimo teórico de la tasa de error verdadero.

Existen varios métodos para estimar la tasa de error de clasificación, los más populares en la literatura se presentan a continuación (Webb, 1999).

Método 1. Error por resubstitución o error aparente

Este es simplemente la proporción de observaciones de la muestra de entrenamiento que son erróneamente clasificados por el clasificador hallado con la misma muestra, se calcula usando la fórmula:

$$e_A = \frac{1}{n}\sum_{i=1}^{n} Q\left(z_i, C\left(\underline{x}_i\right)\right) \tag{2.8}$$

donde para cualquier u y v, $\mathrm{Q}(\mathrm{u}, \mathrm{v}) = 0$ si $u = v$ y 1 si $\mathrm{u} \neq \mathrm{v}$.

Por lo general, el error por resubstitución es un estimador demasiado optimista y sesgado, y puede conducir a falsas conclusiones si el tamaño de la muestra no es muy grande comparado con el número de variables envueltas en el clasificador, ya que en este caso el clasificador modela más el ruido en los datos que su estructura (Webb, 1999).

Método 2. Error usando una muestra de prueba

En este caso, se elige en forma aleatoria una proporción de objetos en la muestra inicial (usualmente entre 80-90%), la cual es usada como muestra de entrenamiento y con la que se construye el clasificador. Los objetos restantes conforman el conjunto de prueba, con el cual se prueba el clasificador, es decir, la tasa de error de clasificación se calcula usando este conjunto. Muchos de los conjuntos disponibles ya traen específicamente su conjunto de prueba y entrenamiento.

Es claro que al usar este método hay una pérdida de eficiencia debido a que no se usan todos los objetos disponibles para construir el clasificador, situación que no es muy grave cuando el tamaño de muestra inicial es de gran tamaño.

Método 3. Error por validación cruzada

En su forma más simple, este método consiste en dividir al azar la muestra en v partes ($v=10$, es el más utilizado). Cada parte constituye una muestra de prueba y con las $(v-1)$ restantes se halla el clasificador, de esta forma se procede a clasificar la muestra de prueba y se calcula el error de clasificación para cada parte. El error de clasificación es el promedio de las clasificaciones erradas obtenidas en cada una de las v partes. De esta forma se obtiene una estimación del error de clasificación, que es poco sesgada, pero muy variable (Webb, 1999).

En la presente investigación se hace uso del método de validación cruzada con $v=10$, para estimar el error de clasificación en los conjuntos de datos pequeños a grandes y del método del conjunto de prueba para los conjuntos más grandes.

2.9. Clasificadores

Entre los métodos clásicos para resolver el problema de clasificación supervisada, que se usa en el trabajo de investigación se tiene: al análisis discriminante lineal (LDA), regresión logística múltiple nominal (LOG), el método de los k-vecinos más cercanos (KNN), SVM y clasificación por árboles (RPART).

2.9.1. Análisis discriminante lineal (LDA)

El análisis discriminante lineal puede derivarse como el método de máxima verosimilitud para poblaciones normales con medias diferentes y matrix de covarianza común, donde cada clase es representada por su centroide y se clasifica a los objetos más cercanos usando una métrica apropiada.

Se asume que la función de densidad condicional del predictor en cada clase, denotada por $P(X/G)$, es gaussiana multivariada con cada clase teniendo su propio vector de medias μ_j, pero compartiendo una matriz de covarianza común Σ ($NM(\mu, \Sigma)$).

Las probabilidades *a priori* son $P(G = j) = \Pi_j$. En la configuración idealizada, donde todo es conocido, se obtiene el clasificador óptimo de Bayes. Si las nuevas observaciones son clasificadas a partir de la misma distribución conjunta, la regla:

$$C(\underline{x}) = j \text{ si } P(G = j / \underline{x}) = \max_l P(G = l / \underline{x})$$

alcanza la tasa mínima de mala clasificación. En este caso se tiene:

$$P(G = j / X = \underline{x}) = \frac{f(\underline{x}; \mu_j, \Sigma)\Pi_j}{\sum_l f(\underline{x}; \mu_l, \Sigma)\Pi_l} \qquad (2.9)$$

2.9.2. Clasificador basado en los vecinos más cercanos (KNN)

El clasificador basado en los k vecinos más cercanos (*k Nearest Neighbor*) (KNN), propuesto por Cover y Hart (1967) es un método no paramétrico de estimación de la función de densidad. Este método es asintóticamente óptimo y dependiente de la escala (métrica) (Jain *et al.*, 2000). El proceso de clasificación usando KNN se realiza de la siguiente forma:

1. Dada una instancia I, primero se hallan las k instancias que están a una distancia más cercana a I, el valor de k es proporcionado por el usuario, usualmente un número impar.
2. Si la mayoría de esos k objetos pertenecen a la clase G, entonces la instancia es considerada que también pertenece a ella. En caso de empate se clasifica la instancia I, como perteneciente a la clase de la instancia más cercana a ella.

2.9.3. Clasificador RPART

Breiman *et al*. (1984) introdujeron el uso de árboles binarios de decisión en el área de estadística y aprendizaje automático. Ellos implementaron nuevos algoritmos para construcción de árboles y los aplicaron a problemas de regresión y clasificación supervisada. El clasificador RPART (PARTicionamiento Recursivo), que se usa en esta investigación es una implementación en R del método conocido como CART (*Classification and Regression Tress*).

2.9.4. Clasificador SVM

Vapnik y Lerner (1963), presentaron por primera vez el clasificador SVM (*Support Vector Machine*), ideada para la resolución de problemas de clasificación binarios en los que las clases eran linealmente separables. Por este motivo se conocía también como "hiperplano óptimo de decisión" ya que la solución proporcionada es aquella en la que se clasifican correctamente todas las muestras disponibles, colocando el hiperplano de separación lo

más lejos posible de todas ellas. Las muestras más próximas al hiperplano óptimo de separación son conocidas como muestras críticas o "vectores soporte", que es lo que da nombre a la SVM.

Sin embargo, los mejores resultados se obtienen usando una SVM no lineal. Para obtener funciones de clasificación no lineales, la SVM busca el hiperplano óptimo de decisión en un espacio en el que previamente se han transformado los datos ("espacio de características") y este hiperplano se convierte en una función de decisión no lineal en el espacio original. La ventaja de esta transformación es que sólo se debe conocer su núcleo reproductor o "kernel" de la transformación, lo cual simplifica en gran medida la obtención de funciones de decisión no lineales.

2.9.5. Clasificador basado en la regresión logística nominal (LOG)

En regresión logística, cada fila de la matriz de variables predictoras corresponde a las observaciones del vector p-dimensional $x = (x_1, x_2, \cdots, x_n)^T$. Las entradas del vector de respuesta Y, corresponden a la observación de la variable, la cual representa una categoría codificada dentro del conjunto $\{1, 2, \cdots, G\}$, que se llamará grupo o clase para efectos de clasificación supervisada. En nuestro caso esta variable es de tipo nominal; debido a que no hay un orden natural en las categorías de la variable respuesta. Aquí una categoría es elegida arbitrariamente como la categoría de referencia. Supongamos que esta es la primera categoría, entonces la probabilidad de clasificar una observación en una de las G clases se obtiene del modelo:

$$\log\left(\frac{P(y=k)}{P(y=1)}\right) = c_k + \beta_{1k}x_1 + \beta_{2k}x_2 + \cdots + \beta_{pk}x_p \quad k = 2, 3, \cdots, G \qquad (2.10)$$

Después de estimar los parámetros de la regresión logística se puede hacer la predicción de una observación $x = (x_1, x_2, \cdots, x_n)^T$, la cual consiste en la clasificación de dicha

observación en una de las G clases. Para lograr este objetivo se estiman las probabilidades de pertenecer a cada una de las G clases y se aplica la siguiente regla:

$$x \in clase\ G^* \Leftrightarrow G^* = \arg\max_k P(y = k) \tag{2.11}$$

2.10. Función distancia

La función distancia (o la función de similaridad), usada para medir qué tan lejanas (o cercanas) están dos instancias, puede tener un efecto considerable sobre los resultados de ciertos sistemas de aprendizaje (Wilson y Martinez, 2000).

Dadas dos instancias I_i y I_j, la función de distancia más común y más utilizada es la distancia euclidiana, la cual se define como:

$$d(I_i, I_j) = \sqrt{\sum_{k=1}^{p} (I_{ik} - I_{jk})^2} \tag{2.12}$$

Donde p es el número de variables. Esta medida es apropiada cuando todas las variables son numéricas y tienen aproximadamente el mismo rango de valores. Si las variables no tienen el mismo rango de valores la alternativa es usar la distancia normalizada, esto es, dividir las distancias entre las variables por el rango o la desviación estándar de la variable (Wilson y Martinez, 2000).

Cuando se tienen variables nominales, algunos autores definen la distancia entre dos valores de la variable como 0 si los valores son iguales y 1 si los valores son diferentes. Una función distancia alternativa para valores nominales, es la definida por Stanfill y Walz (1986), VDM (Value Difference Metric). Usando VDM, la distancia entre los valores x, y de una variable nominal w, está dada por:

$$VDM_w(x, y) = \sum_{c=1}^{C} \left(\frac{N_{w,x,c}}{N_{w,x}} - \frac{N_{w,y,c}}{N_{w,y}} \right)^2 \tag{2.13}$$

donde $N_{w,x}$ es el número de veces que la variable w tiene el valor x; $N_{w,x,c}$ es el número de veces que la variable w tiene el valor de x y la clase es c; C es el número de clases.

En la parte experimental, de este trabajo se hace uso de la función distancia que combina variables numéricas y nominales, denominada HVDM (*Heterogeneous Value Distance Metric*) (Wilson y Martinez, 1997), la cual es definida como:

$$HVDM(x,y) = \sqrt{\sum_{w=1}^{p2} d_w^2(x_w, y_w)} \tag{2.14}$$

donde $d_w(x,y)$, es la función que define la distancia para la variable w, y está definida como:

$$d_w(x,y) = \begin{cases} VDM_w(x,y), & \text{si } w \text{ es nominal} \\ \dfrac{|x-y|}{Rango(w)}, & \text{si } w \text{ es numérica} \end{cases} \tag{2.15}$$

2.11. Implementación de programas

Para la realización de este trabajo de investigación, fue necesaria la implementación de diversas funciones en el lenguaje R (R Development Core Team, 2006), utilizando algunas de las librerías disponibles (Venables y Ripley, 2002). Estas funciones fueron integradas y permitieron llevar a cabo los cálculos necesarios para realizar las distintas comparaciones que fueron propuestas, es decir, funciones que permitan estimar el error de clasificación utilizando el método de validación cruzada para los clasificadores: LDA (librería MASS), KNN (librería class), RPART (librería rpart), SVM (librería e1071) y LOG (librería nnet). De la misma forma se implementaron funciones para calcular las medidas de complejidad, la detección de ruido y la selección de variables e instancias, para lo cual también se utilizaron como funciones auxiliares algunas de las funciones de la librería dprep.

La implementación de las funciones usadas para este estudio se realizó utilizando programas en R, extendiendo el código de R con código compilado en C++, para realizar esta interfase de R con C++, se utilizó la librería Rcpp. La elaboración de las funciones y programas en C++ se realizó utilizando la librería STL (*Standard Template Library*) y para algunas funciones que requerían de la búsqueda de los k vecinos más cercanos se usó la librería ANN (*Approximate Nearest Neighbor*) (Mount y Arya, 1997).

El software utilizado es el programa estadístico R versión 2.4.1. en su versión para Windows XP. Los programas se ejecutaron usando una computadora Workstation DELL® PrecisionTM 690, el cual tiene un procesador Intel® XeonTM con una velocidad de 3.00 GHz, bajo una arquitectura de 64 Bits y con 16 GB de memoria RAM.

Más detalles de la implementación de las funciones, se encontrarán en las mismas funciones implementadas, las cuales están disponibles en la página Web del grupo CASTLE del departamento de Matemáticas de la Universidad de Puerto Rico Recinto de Mayagüez: http://academic.uprm.edu/~eacuna.

Capítulo III

Complejidad de los Datos en Problemas de Clasificación Supervisada

3.1. Introducción

Como se ha mencionado en el capítulo anterior, una tarea fundamental en la minería de datos es el reconocimiento de patrones escondidos en un conjunto o base de datos. Este es un problema de análisis multivariado conocido como clasificación y tiene dos variantes, la clasificación supervisada y la clasificación no supervisada. El desarrollo de la presente investigación trata los problemas de clasificación supervisada.

El análisis discriminante o clasificación supervisada está con la asignación de nuevas instancias a grupos previamente definidos, haciendo uso de la información proporcionada por un conjunto de variables predictoras. Esto significa que existe una relación entre una variable categórica, que determinan los grupos y un conjunto de variables interrelacionadas, denominadas predictoras.

Explícitamente, supóngase que existe un número finito de grupos (clases o categorías), conocidas *a priori*, $G_1, G_2, \cdots, G_g$; un objeto de interés que se asume pertenece a uno y sólo uno de los grupos, y cuya pertenencia se denota por z, es decir, $z = i$, significa que el objeto pertenece al grupo G_i $(i = 1, 2, \cdots, g)$. Además se tiene el vector p-dimensional, $\underline{x} = (x_1, x_2, \cdots, x_p)$, que representa las mediciones correspondientes a p características disponibles para cada objeto en estudio. De acuerdo con este contexto, el análisis

discriminante estudia la relación entre una variable categórica, que definen los grupos y el vector de características.

Desde el punto de vista de la teoría de decisión, la pertenencia de un nuevo objeto a un grupo es desconocida y el problema a resolver es hacer una asignación adecuada del objeto a uno de los g grupos basándose en las mediciones de sus p características asociadas, proceso que se realiza construyendo un clasificador o regla de clasificación (McLachan, 1992).

En términos simples, se establecen reglas de clasificación a partir de un conjunto de instancias seleccionadas al azar, denominado conjunto de entrenamiento (E), y se aplican para clasificar instancias en un conjunto de prueba para evaluar la exactitud de la clasificación. En este contexto, el rendimiento de cada clasificador está fuertemente relacionado con las características de los datos. Por consiguiente, un análisis de características de los datos parece ser una herramienta esencial para seleccionar el algoritmo de clasificación apropiado a un problema particular.

El objetivo de los problemas de clasificación supervisada es el de minimizar la tasa de error de clasificación. Teóricamente, si se desea de manera precisa medir la complejidad de un problema de clasificación, la mejor medida es calcular el error de Bayes, ε, que por definición , es la tasa mínima de error de clasificación que se pueda obtener (Duda *et al*., 2000, Fukunaga, 1990). Lamentablemente, el cálculo directo del error de Bayes, en la práctica es muy difícil de obtener.

Varios estudios han mostrado que el comportamiento empírico de los clasificadores está fuertemente relacionado a los datos disponibles (Attoor y Dougherty, 2004). *A priori*, es muy difícil determinar que clasificador tendrá un mejor rendimiento, dado un problema específico. Generalmente, se han realizado estudios experimentales con varios clasificadores, y se han reportado los errores de clasificación en un número limitado de problemas, sin analizarse el porqué de las diferencias en el rendimiento de los

clasificadores, para cada problema de clasificación. En este contexto, resulta razonable determinar la complejidad de un problema de clasificación antes de usar los algoritmos de clasificación existentes. La estimación de la complejidad nos puede permitir predecir el comportamiento de los clasificadores individuales o combinados y realizar selección de variables y/o instancias para un problema de clasificación en particular.

Recientemente, se han realizado estudios tendientes a explicar el comportamiento de los clasificadores, relacionando su rendimiento con la estructura de los datos utilizados (Ho y Bernadó, 2006; Bernadó y Ho, 2004; Ho, 2001, 2002; Lee y Hwang, 1996 y Sohn, 1999). El enfoque general consiste en predecir la aplicabilidad y rendimiento de un clasificador a base de ciertas características de los datos. Con este fin, podemos emplear un conjunto de medidas de la complejidad de los datos. Por ejemplo, Sohn (1999) obtiene un total de 19 características de los datos y realiza un análisis de regresión entre la proporción del error de once clasificadores (incluyendo clasificadores estadísticos, de aprendizaje automático y redes neuronales) y las medidas de los conjuntos de los datos.

Bernardó y Ho (2004) definen nueve medidas de complejidad de los datos, miden y calculan las medidas de complejidad para cada problema de dos clases usando toda la información disponible. Ellos buscan las regiones en el espacio de complejidad donde cada clasificador es significativamente mejor que otros y las regiones donde los métodos de la clasificación múltiples rindan de manera similar.

En otros trabajos, Bernadó y Ho (2005) y Bernadó *et al.*, (2006), se investiga el dominio y la capacidad de un sistema de clasificadores basados en algoritmos genéticos, denominado XCS (Wilson, 1998), mediante un conjunto de medidas de complejidad geométrica. Ellos conducen un estudio de 392 problemas de la clasificación (con dos clases), junto a la caracterización de la complejidad, buscando identificar los dominios difíciles y fáciles para XCS. Los resultados obtenidos demuestran altas correlaciones entre el funcionamiento de XCS's y las medidas de la longitud de las fronteras y la compacidad de las clases, como también con la no linealidad de las fronteras de decisión.

Mollineda *et al.*, (2005) investigan la aplicación de un conjunto de medidas de complejidad como una herramienta que permita predecir el rendimiento de un algoritmo de selección de prototipos en un problema particular. Ellos prueban diferentes medidas de la complejidad de los datos usando 17 conjuntos de datos, derivando una serie de situaciones prácticas bajo las cuales determinado método de selección de prototipos actúa adecuadamente.

Ho y Basu (2000, 2002) presentan algunos factores que pueden contribuir a la complejidad de un problema de clasificación:

a) **La ambigüedad de la clase**. Algunos problemas de clasificación pueden tener clases que no son discernibles. Puede haber por ejemplo, instancias que pertenecen a clases diferentes y que tienen los mismos valores en las variables. Para resolver este problema se requiere de conjuntos de datos con un número suficientemente grande de instancias que puedan reflejar las distribuciones de probabilidad *a priori* y *posteriori* de las clases. En estos casos los problemas de clasificación son reconocidos por tener un error de Bayes diferente de cero, debido a que con la información disponible no es posible que el clasificador trabaje adecuadamente. Esto se debe a la ambigüedad intrínsica del problema, o al hecho que las variables no son suficientes para distinguir una clase de la otra. En este último caso, el efecto puede aliviarse redefiniendo o aumentando el conjunto de variables.

b) **Tamaño y dimensión del conjunto entrenamiento**. El número y la representatividad de las instancias disponibles en el conjunto de entrenamiento condicionan la capacidad de generalización del clasificador.

c) **Complejidad de la frontera de decisión.** La complejidad de las fronteras de decisión, constituye otro factor que condiciona el rendimiento de un clasificador.

Los errores de clasificación muy altos se pueden deber a la combinación de muchos factores. Por lo cual, si se desea mejorar el rendimiento de los clasificadores se tienen que tratar con cada uno de ellos de alguna forma. Por ejemplo, la ambigüedad de las clases, o es una naturaleza del problema o el problema requiere que se incluyan más variables (Ho y Basu, 2000).

3.2. Medidas para la caracterización de datos

Una forma simple de cuantificar la complejidad de un problema de clasificación es calcular la tasa de error de clasificación para un clasificador elegido. Sin embargo, si lo que se quiere es estudiar el comportamiento de los clasificadores, se tienen que usar otras medidas que sean independientes de la elección del clasificador pero capaces de discernir su competencia. Como hemos mencionado anteriormente, el comportamiento de los clasificadores tiene una fuerte dependencia de la complejidad (estructura) de los datos. Teóricamente, se han buscado cotas límites de exactitud, para el error de clasificación o rendimiento, basándose en condiciones y/o supuestos teóricos que no se ajustan a la realidad. Por otro lado, el análisis empírico se ha basado normalmente en la medición de la exactitud (o los errores de clasificación) del clasificador en un número pequeño de conjunto de datos, sin establecerse el porqué de las diferencias en los comportamientos de los clasificadores.

Los estudios teóricos y empíricos usualmente ignoran las particulares descripciones estadísticas y geométricas de las distribuciones de los datos para cada clase, y las fronteras de decisión, para explicar los resultados obtenidos en la clasificación de los conjuntos de datos. En varios artículos recientes (Bernadó y Ho, 2004; Ho, 2002; Singh, 2003a, 2003b y Sohn, 1999), se introducen el uso de medidas para caracterizar la complejidad de los datos y relacionar tal descripción al rendimiento del clasificador.

La mayoría de las medidas de complejidad de los datos propuestos y utilizados por diversos autores sólo se definen para problemas de clasificación de dos clases y están enfocadas principalmente en la efectividad de una sola variable (Bernadó y Ho., 2004; Ho,

2002; Ho y Basu, 2002; Singh, 2003a, 2003b). En este trabajo se tratarán problemas de clasificación con g clases y se procederá a usar medidas generalizadas para problemas de más de dos clases considerando la información proporcionada por todas las variables relevantes. En las siguientes secciones se describen varias medidas seleccionadas de varios artículos y nuestras medidas propuestas para cuantificar la complejidad del problema de clasificación.

3.2.1. Razón discriminante de Fisher's (F1)

La versión más simple de esta medida es muy conocida y estima cuán separables están dos clases mediante una variable específica (Ho, 2001, 2002)

$$F_1 = \frac{(\mu_1 - \mu_2)}{\sigma_1^2 + \sigma_2^2} \tag{3.1}$$

donde μ_1, μ_2, σ_1^2 y σ_2^2 son las medias de las dos clases y sus variancias, respectivamente. En un problema de clasificación multidimensional se considera el máximo de los valores obtenidos para la razón discriminante usando cada una de las variables.

Otra posible generalización para g clases y que también considera todas las variables, es la siguiente (Mollineda *et al*., 2005; Sotoca *et al*., 2005):

$$F1_{gen} = \frac{\sum_{i=1}^{g} n_i \delta(\mu, \mu_i)}{\sum_{i=1}^{g} \sum_{j=1}^{n_i} \delta(x_j^i, \mu_i)} \tag{3.2}$$

donde n_i denota el número de instancias en la clase i, δ es una métrica, μ es la media global, μ_i; es la media de clase i, y x_j^i representa la instancia j que pertenece a la clase i.

3.2.2. Volumen de la región de traslapo (*overlap*, F2)

Esta medida calcula, para cada variable f_k, la longitud del traslapo normalizado por la longitud del rango total en que todos los valores de ambas clases están distribuidos (Ho, 2001, 2002)

$$F2 = \prod_k \frac{\min \max_k - \max \min_k}{\max \max_k - \min \min_k} \tag{3.3}$$

donde $k = 1, 2, \cdots, d$, para un problema de dimensión d, y

$$\min \max_k = \min\left\{\max\left(f_k, c_1\right), \max\left(f_k, c_2\right)\right\}$$

$$\max \min_k = \max\left\{\min\left(f_k, c_1\right), \min\left(f_k, c_2\right)\right\}$$

$$\max \max_k = \max\left\{\max\left(f_k, c_1\right), \max\left(f_k, c_2\right)\right\}$$

$$\min \min_k = \min\left\{\min\left(f_k, c_1\right), \min\left(f_k, c_2\right)\right\}$$

Una generalización muy simple de la medida F2 para el problema del g clases puede obtenerse sumando la medida para todos los posibles pares de clases (Mollineda, *et al*., 2005; Sotoca *et al*., 2005):

$$F2 = \sum_{(c_i, c_j)} \prod_k \frac{\min \max_k - \max \min_k}{\max \max_k - \min \min_k} \tag{3.4}$$

donde $\left(C_i, C_j\right)$ recorre por todos los pares de clases, $i, j = 1, 2, \cdots, g$, y:

$$\min \max_k = \min\left\{\max\left(f_k, c_i\right), \max\left(f_k, c_j\right)\right\}$$

$$\max \min_k = \max\left\{\min\left(f_k, c_i\right), \min\left(f_k, c_j\right)\right\}$$

$$\max \max_k = \max\left\{\max\left(f_k, c_i\right), \max\left(f_k, c_j\right)\right\}$$

$$\min \min_k = \min\left\{\min\left(f_k, c_i\right), \min\left(f_k, c_j\right)\right\}$$

3.2.3. Fracción de puntos en la frontera (F3)

Este método se basa en el uso de *Minimum Spanning Tree* (MST) propuesto por Friedman y Rafsky (1979). El método construye un MST que conecta todas las instancias en el conjunto de entrenamiento a su vecino más cercano, sin considerar la clase a la que pertenece. Luego se procede a contar el número de instancias conectadas a una instancia de la clase contraria por una arista en el MST. Se considera que estos puntos están cerca de la frontera de la clase. La fracción de las instancias restantes sobre el número total de instancias en el conjunto de entrenamiento se usa como una medida de complejidad (F5) (Jain *et al*., 2002).

3.2.4. Separabilidad lineal (F4, F5)

La separabilidad lineal es la máxima probabilidad de clasificación correcta cuando se usan hiperplanos para discriminar la distribución de patrones. En problemas de dos clases, representa la probabilidad de solapar si cada clase es distribuida en una región convexa.

Los clasificadores lineales pueden ser obtenidos por una formulación de programación lineal propuesta por Smith (1968) que minimiza la suma de distancias de los puntos de error al hiperplano de separación (substrayendo un margen constante).

$$\begin{aligned} \min \quad & a^t t \\ s.t. \quad & Z^t w + t \geq b \\ & t \geq 0 \end{aligned} \tag{3.5}$$

donde a, b son los vectores constantes arbitrarios, w es el vector de peso, t es un vector del error, y Z es una matriz donde cada columna z se define en un vector de la entrada x y su clase G (con el valor c_1 o c_2) como sigue:

$$\begin{aligned} z &= +x \quad si\ c = c_1 \\ z &= -x \quad si\ c = c_2 \end{aligned}$$

El valor de la función objetivo es la que usan Ho y Basu (2002), como una medida de la separabilidad de las clases (F4), es cero para un problema linealmente separable.
Por otro lado, una segunda medida (F5) simplemente corresponde al error de clasificación de un clasificador lineal en el conjunto de entrenamiento original.

3.2.5. Separabilidad no paramétrica de clases (F6, F7)

La primera medida (F6) es la razón de las distancias promedio a los vecinos más cercanos en su misma clase y la distancia promedio a los vecinos más cercanos en la otra clase. Con esta medida de complejidad se compara la dispersión intra-clase con la separabilidad entre clases. Los valores más pequeños hacen pensar en más datos discriminantes. La segunda medida (F7) simplemente corresponde a la estimación del error de clasificación usando el clasificador basado en los vecinos más cercanos por el método dejando uno afuera.

3.2.6. E-vecindad (F8)

Esta medida cuenta el número de bolas que se necesitan para cubrir cada clase, mientras se centra cada bola en un punto del conjunto de entrenamiento y crece al tamaño máximo antes de que alcance un punto de otra clase (Ho y Basu, 2002). Las bolas redundantes que quedan completamente en el interior de otras bolas son descartadas. Esta cuenta se normaliza por el número total de puntos.
Esta medida proporciona una descripción interior en lugar de una descripción de la frontera como las medidas basadas en MST (ver Sección 3.2.3).

3.2.7. Promedio de puntos por dimensión (F9)

Una medida que podría contribuir a explicar el comportamiento de algunos problemas de la clasificación es calcular la medida F9 que describe la densidad de las distribuciones espaciales de las muestras calculando el número de instancias en el conjunto de datos sobre el número de variables en el estudio.

Para esta medida, vamos a considerar la estimación calculando el número de instancias en el conjunto de entrenamiento sobre el producto del número de clases y el número de variables.

3.2.8. Medidas de distancia probabilística

Teóricamente, la mejor estimación para describir la separabilidad de dos a más clases, es el error de Bayes (Fukunaga, 1990). Sin embargo, en la práctica es difícil de estimar debido a su complejidad computacional, por lo que se obtiene de manera empírica en lugar de una derivación analítica. Ante esta situación, existe un número de distancias estadísticas-probabilísticas que proporcionan cotas superiores para el error mínimo de separabilidad en problemas de clasificación de dos clases. Debido a que los datos reales no necesariamente son normales, los resultados obtenidos usando estas medidas deben de ser tratados con precaución. Algunas de estas medidas son: Bhattacharya, Chernoff, Divergencia, Mahalanobis y Matusita.

3.2.9. Medidas estadísticas

En el proyecto Statlog (King *et al*., 1995; Michie *et al*., 1994) se compararon varias técnicas de clasificación utilizando 22 conjuntos de datos. Estos conjuntos se describen en términos de varias estadísticas intentando predecir la pertinencia de un clasificador basado en ciertas características de los datos.

Entre otras, las siguientes medidas estadísticas descriptivas y multivariantes, se utilizan para resumir los datos en el proyecto Statlog: número total de instancias, número de instancias en el conjunto de entrenamiento, número de instancias en el conjunto de prueba, número de variables, número de atributos binarios, número de clases, los coeficientes de correlación, media absoluta entre dos variables, sesgo promedio de las características, curtosis media de las variables, entropía de las clases, y la entropía promedio de las variables discretas.

En la siguiente sección se describe el problema de clasificación y en ese contexto se proponen y describen dos medidas de la calidad de las instancias del conjunto de

entrenamiento. Adicionalmente se plantean dos medidas de complejidad de los problemas de clasificación basada en la calidad de los datos.

3.3. Identificación del problema de clasificación

Siguiendo la notación dada por McLachan (1992), sea *X* el vector aleatorio p-dimensional de características, cuya observación o medición de las *p* características en el objeto en consideración se denota por $\underline{x}$, la variable asociada *z*, que denota al grupo de origen del objeto, se reemplaza por un vector *g*-dimensional *z* de variables indicadoras, donde, la *i*-ésima componente de z está definida como uno o cero de acuerdo a la pertenencia del objeto al *i*-ésimo grupo G_i $(i = 1, 2, \cdots, g)$, es decir,

$$z_i = \begin{cases} 1 & \underline{x} \in G_i \\ 0 & \underline{x} \notin G_i \end{cases} \qquad \forall\, i = 1, 2, \cdots, g\,. \tag{3.6}$$

En el análisis discriminante se asume que la probabilidad *a priori* de que un objeto pertenezca al grupo G_i es π_i, esto es:

$$\pi_i = \Pr[Z_i = 1], \text{ y } \pi_i \geq 0 \ \ (i = 1, 2, \cdots, g) \tag{3.7}$$

Sea $f(\underline{x}/G_i)$ o $f_i(\underline{x})$ la función de densidad (condicional) de $\underline{x}$ en la clase G_i, la probabilidad *a posteriori* de que el objeto con vector de mediciones $\underline{x}$ (esto es, $X = \underline{x}$), pertenezca al grupo G_i, aplicando el teorema de Bayes, está dada por:

$$\begin{aligned} \Pr[\text{el objeto} \in G_i / \underline{x}] &= \Pr[Z_i = 1/\underline{x}] \\ \Pr[\text{el objeto} \in G_i / \underline{x}] &= \frac{\Pr[Z_i = 1, X = \underline{x}]}{f(\underline{x})} \\ \Pr[\textit{el objeto} \in G_i / \underline{x}] &= \frac{\Pr[Z_i = 1] f(\underline{x}/Z_i = 1)}{f(\underline{x})} \end{aligned} \tag{3.8}$$

$$\Pr[\text{el objeto} \in G_i / \underline{x}] = \frac{\pi_i f_i(\underline{x})}{\sum_{i=1}^{g} \pi_i f_i(\underline{x})}$$

Una regla de clasificación o clasificador se denotará como $C(\underline{x})$, donde $C(\underline{x}) = i$ significa que un objeto con vector de características $\underline{x}$ es asignado al i-ésimo grupo G_i $(i = 1, 2, \cdots, g)$. Más explícitamente, esta regla lo que intenta hacer es una partición del espacio de características en g regiones mutuamente excluyentes $R_1, R_2, \cdots, R_g$ de tal forma que si $\underline{x}$ cae en la región R_i, el objeto con vector de mediciones $\underline{x}$ se asigna al grupo G_i $(i = 1, 2, \cdots, g)$.

De acuerdo con estas definiciones, una instancia será mal clasificada cuando cae en una región que no está etiquetada con la misma clase a la que pertenece. Conceptualmente, la mala clasificación ocurre en las regiones del espacio de características, para los cuales, dados los valores observados, se traslapan dos o más clases. Por consiguiente, las regiones traslapadas corresponden a la intersección de dos o más clases (ver figura 3.1). La estimación de estas regiones traslapadas, constituye una medida de la complejidad del problema de clasificación asociado, ya que precisamente, estas regiones van a definir las características de la frontera de decisión y la separabilidad de las clases. En la siguiente sección definimos las medidas de calidad de las instancias que pertenecen a cada grupo. Los valores asignados a las instancias de acuerdo a su calidad en cuanto a su ubicación espacial, nos van a permitir cuantificar la complejidad de los problemas de clasificación, dado un conjunto de datos.

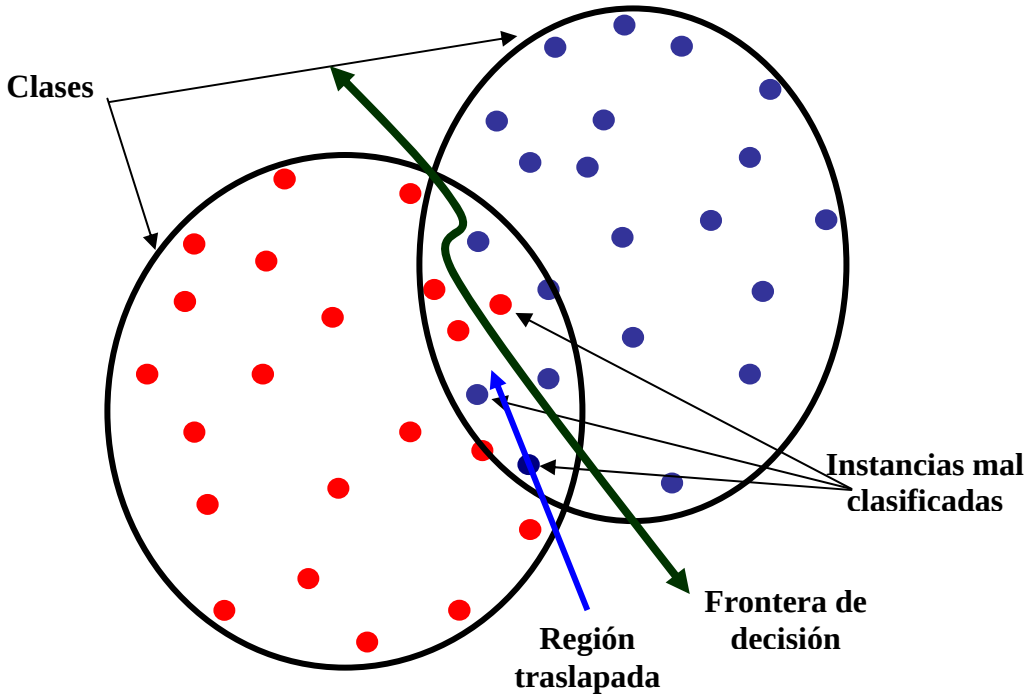

Figura 3.1. Representación gráfica de las regiones traslapadas, mala clasificación y separabilidad de clases

3.4. Medidas de la calidad de las instancias

En esta investigación se propone el uso de dos medidas de calidad de las instancias en cada clase. Las medidas caracterizan a las instancias con respecto a su ubicación dentro de su clase y con respecto a las fronteras de decisión.

3.4.1. Calidad de las instancias con respecto a los centroides (Q)

Esta medida de calidad de las instancias indica qué tan cerca está una instancia determinada del centro de la clase a la que pertenece con respecto a las otras clases. De acuerdo con los valores que se obtengan para la medida podemos determinar si una instancia es del centro de la clase, está ubicada en el borde de la clase o es una instancia

que puede ser considerada como ruido. El procedimiento de cálculo para determinar la calidad de las instancias es como sigue:

Dado un conjunto de entrenamiento E con n instancias, para cada instancia I_i en E que pertenece a una de las g clases $\{G_1, G_2, \cdots, G_g\}$, se calcula la siguiente medida de calidad de la instancia:

$$Q_i = \frac{r_i - d_i}{\max(d_i, r_i)} \tag{3.9}$$

donde: d_i es la distancia métrica de la i-ésima instancia (x_i) al centroide de su misma clase; y r_i es la distancia métrica mínima de la i-ésima instancia a los centroides de las clases opuestas; normalizada por el máximo de d y r.

La medida de calidad con respecto a los centroides (Q_i) tiene las siguientes propiedades:

1. Q_i varía entre -1 y 1.
2. Un valor de Q_i=1 o cercano a 1, indica que es una instancia de buena calidad, o ubicada cerca del centro de su clase.
3. Un valor de Q_i = -1, o cercana a -1, indica una instancia ruidosa o de baja calidad.
4. Las instancias ubicadas en la frontera de decisión van a tener valores cercanos a cero.

Como se puede observar, la medida de calidad de las instancias constituye un buen indicador para identificar instancias ubicadas en el centro de su clase, la frontera de decisión o si es una instancia ruidosa.

3.4.2. Calidad de las instancias basada en los vecinos más cercanos (QNN)

Alternativamente a la medida anterior Q, proponemos una medida de calidad basada en los vecinos más cercanos (NN), que la denotaremos por QNN.

A diferencia de la medida de calidad con respecto a los centroides, esta medida de calidad de las instancias indica qué tan cerca (o qué tan lejos), está una instancia determinada de la frontera de decisión entre la clase a la que pertenece y la clase más cercana a ella. Sin embargo, esta medida utilizada para detectar las instancias en la frontera de decisión se va a ver afectada por la presencia de ruido en las clases. Una forma de resolver este problema es detectar y eliminar el ruido en las clases y luego utilizar la medida para identificar las instancias en la frontera de decisión. De acuerdo con los valores que se obtengan para la medida podemos determinar si una instancia está cerca de la frontera de decisión o en un análisis del ruido en las clases, determinar si la instancia en particular es ruido. El procedimiento de cálculo para determinar la calidad de las instancias basadas en los vecinos más cercanos, es como sigue:

Dado un conjunto de entrenamiento E con n instancias, para cada instancia I_i en E que pertenece a una de las g clases $\{G_1, G_2, \cdots, G_g\}$, se calcula la siguiente medida de calidad de la instancia:

$$QNN_i = \frac{\tilde{r}_i - d_i}{\max\left(\tilde{r}_i, d_i\right)} \tag{3.10}$$

donde: d_i es la distancia métrica de la *i*-ésima instancia (I_i) a su vecino más cercano en su misma clase; $\tilde{r}_i$ es la distancia métrica mínima de la *i*-ésima instancia a sus vecinos más cercanos de cada una de las clases opuestas, normalizada por el máximo de d y $\tilde{r}$.

La medida de calidad usando los vecinos más cercanos (QNN_i) tiene las siguientes propiedades:

1. QNN_i varía entre -1 y 1.
2. Un valor de QNN_i =1 o cercano a 1, indica que es una instancia que se encuentra bastante alejada de la frontera de decisión.
3. Un valor de QNN_i = -1, o cercana a -1, indica una instancia ruidosa o de baja calidad.
4. Las instancias ubicadas en la frontera de decisión van a tener valores cercanos a cero.

En este caso, se puede observar que la medida de calidad de las instancias (QNN), constituye un buen indicador para identificar en un primer análisis las instancias ruidosas, y posteriormente las instancias que están cerca de la frontera de decisión.

Una manera de mejorar la robustez de la medida de calidad QNN a la presencia de ruido en las clases es usar el promedio de las distancias a los k vecinos más cercanos de la instancia I_i como medida para d_i y la distancia mínima promedio a los k vecinos más cercanos de cada una de las clases opuestas.

3.5. Medidas de complejidad basadas en la calidad de las instancias

En esta investigación se propone el uso de dos medidas de calidad de las instancias en cada clase. Las medidas caracterizan a las instancias con respecto a su ubicación dentro de su clase y con respecto a las fronteras de decisión.

Para usar las medidas de calidad de las instancias en determinar la complejidad de un conjunto de datos procedemos como sigue:

1. Para cada instancia del conjunto de entrenamiento, se calcula la medida de calidad Q (o QNN).
2. Se procede a contar las instancias que tienen una medida de calidad menor que cero (negativa). La medida reporta la proporción de instancias con "mala" calidad o ruido, respecto al total de instancias en el conjunto de entrenamiento (*n*):

$$Q_{Ind} = \frac{count\{Q_i < 0\}}{n} \tag{3.11}$$

Complejidad computacional. La complejidad en tiempo de computación de las medidas de complejidad basadas en las medidas de calidad de las instancias, está determinada por los pasos necesarios para calcular la medida de calidad Q. En este caso, primero se procede a calcular los centroides para las *g* clases, que son el promedio de los valores para las *p* variables en cada clase, de esta forma el número de pasos necesarios es:

$$\sum_{j=1}^{g} n_j p = Np$$

donde, n_j es el número de instancias en la *j*-ésima clase y *p* el número de variables.

Luego, para cada instancia se procede a calcular su distancia a los centroides, lo cual se realiza en *p* pasos, para un total de Np pasos. Finalmente el cálculo de la medida de complejidad (expresión 3.11), se hace en *N* pasos. El número total de pasos para calcular la medida de complejidad basada en Q es: $Np + Np + N = 2Np + N$. Por consiguiente, la complejidad asintótica en tiempo de computación en términos del número de instancias N y el número de variables p es: $O(Np)$.

Para el caso de las medidas de complejidad usando la calidad de las instancias basada en los *k* vecinos más cercanos QNN, el cálculo de los valores $\tilde{r}$ y $\tilde{d}$, se hace en el peor de los casos en $N^2 p$ pasos, ya que se tienen que almacenar las distancias entre las N instancias,

que se calculan en p pasos. Por lo tanto, la complejidad asintótica en tiempo de computación para esta medida es de $O\left(N^2 p\right)$

3.6. Medida de traslapo (OVER)

Adicionalmente, a las medidas de calidad de las instancias, proponemos la siguiente medida para calcular el traslapo (*overlap*), en problemas de clasificación supervisada. Dado un conjunto de entrenamiento E con *n* instancias, para cada una de las dos clases $\left\{G_1, G_2\right\}$, se calcula la distancia promedio de cada instancia al centro de su clase:

$$R_{G_1} = \left(\frac{\sum_{x \in G_1} d\left(x, C_{G_1}\right)}{n_{G_1}} \right) \qquad R_{G_2} = \left(\frac{\sum_{x \in G_2} d\left(x, C_{G_2}\right)}{n_{G_2}} \right) \tag{3.12}$$

Luego, la medida de separabilidad se calcula, dividiendo la suma de las distancias promedios por la distancia de los centroides entre las clases { G1 ,G2}, de esta forma:

$$Overlap = \frac{\left(\frac{\sum_{x \in G_1} d\left(x, C_{G_1}\right)}{n_{G_1}} \right) + \left(\frac{\sum_{x \in G_2} d\left(x, C_{G_2}\right)}{n_{G_2}} \right)}{d\left(C_{G_1}, C_{G_2}\right)} \tag{3.13}$$

Podemos usar una generalización muy simple de la medida de traslapo para el problema de G-clases, promediando la medida para todos los posibles pares de clases:

$$Overlap_{Gen.} = \sum_{(G_i, G_j)} \frac{\left(\frac{\sum_{x \in G_i} d\left(x, C_{G_i}\right)}{n_{G_i}} \right) + \left(\frac{\sum_{x \in G_j} d\left(x, C_{G_j}\right)}{n_{G_j}} \right)}{d\left(C_{G_j}, C_{G_j}\right)} \tag{3.14}$$

3.7. Resultados Experimentales

3.7.1. Metodología

Para probar y comparar el rendimiento de las medidas propuestas se utilizaron un total de 16 conjuntos de datos obtenidos de la Universidad de California en Invine (UCI): *Abalone, Balance, Bupa, Breastw, Census, Diabetes, Ionosfera, Iris, Landsat, Letter, Penbased, Shuttle, Sonar, Vehicle y Waveform* (ver anexo A). Para cada conjunto de datos se calculan los errores de validación cruzada correspondiente a los cinco clasificadores considerados en esta experimentación, los cuales son: el análisis discriminante lineal (LDA), el clasificador basado en los k vecinos más cercanos (KNN), el clasificador RPART, el clasificador SVM, y el clasificador basado en la regresión logística (LOG), (ver capítulo 1). También se calculan las tres medidas de complejidad propuestas (Q, OVER y QNN), y las medidas de complejidad: Razón discriminante de Fisher's (F1), el volumen de la región de traslapo (*overlap*) (F2) y la Fracción de puntos en la frontera usando MST (F3). Adicionalmente, se calcula la medida Qnorm, la cual es similar a la medida Q, con la única variante de que para los cálculos se usan las distancias métricas normalizadas.

Para determinar la eficiencia relativa de las medidas de complejidad se calcula la correlación entre los errores de clasificación y las medidas de complejidad. La medida de complejidad de los datos que tenga una mayor correlación con los errores de clasificación es más representativa de la complejidad del problema de clasificación.

3.7.2. Resultados Experimentales

La tabla 3.1, muestra los resultados de los errores de clasificación para los dieciséis conjuntos de datos usando validación cruzada, usando los cinco clasificadores. Por otro lado, la tabla 3.2 contiene los resultados experimentales obtenidos para las siete medidas de complejidad. La lectura de los resultados, debe hacerse de la siguiente manera: para las medidas F1, F2, F3 un mayor valor significa una menor complejidad en el problema de clasificación (relación inversa). Para las medidas Q, QNN y OVER, un mayor valor, representa una mayor complejidad para la tarea de clasificación (relación directa).

De acuerdo con la medida de complejidad Q, los conjuntos de datos más complejos para la clasificación supervisada son: *Vehicle, Letter, Abalone, Bupa, Diabetes y Sonar* (en ese orden) y los problemas más sencillos son: *Breastw* e *Iris* (ver tabla 3.2.). Para la medida Qnorm, los conjuntos de datos más complejos para la clasificación supervisada son: *Letter, Vehicle, Abalone y Bupa* (en ese orden) y los problemas más sencillos son*: Breastw, Iris y segment.* Por otro lado, la medida de complejidad QNN, indica que los conjuntos de datos más complejos para la clasificación supervisada son: *Abalone, Bupa, Vehicle y Diabetes* (en ese orden) y los problemas más sencillos son: *Shuttle, Penbased, Breastw, Segment, Letter, Iris y Landsat*. Finalmente, para la medida OVER, los conjuntos de datos más complejos para la clasificación supervisada son: *Bupa Vehicle, Sonar, Diabetes, Abalone e Ionosfera* (en ese orden) y los problemas más sencillos son: *Iris, Breastw y Landsat*.

En los resultados se observan ciertas contradicciones entre los valores de las medidas de complejidad usados para determinar si un conjunto de datos es complejo o no. Podemos ver que el conjunto *Letter*, aparece como un conjunto bastante complejo cuando se usan las medidas Q, Qnorm y OVER. Mientras que por otro lado la medida QNN lo cataloga como un conjunto de complejidad sencilla. La justificación en este caso en particular, es que el conjunto *Letter* es un conjunto cuya frontera de decisión es no lineal, y las medidas Q, Qnorm y Over resaltan como buenas instancias a las que se encuentran ubicadas más cerca del centro de cada clase y las clases que son separables linealmente, por ende, en los conjuntos catalogados como más complejos, los clasificadores lineales no van a realizar un buen trabajo.

Por otro lado, se tiene el caso del conjunto de datos *Vehicle*, en el cual las medidas de complejidad Q, Qnorm y Over sobreestiman su complejidad. La razón para que esto ocurra, es que los centros de las clases del conjunto *Vehicle* se encuentran bastante cercanas y por la naturaleza de las medidas mencionadas, éstas son sensibles a la proximidad de los centroides de las clases, provocando una sobreestimación de la complejidad del conjunto de datos.

Tabla 3.1. Tasas de error de clasificación usando validación cruzada para los diferentes conjuntos de datos

CONJUNTO	LDA	KNN	SVM	RPART	LOG
ABALONE	36.12	38.82	34.15	37.62	35.09
BALANCE	13.28	13.62	9.57	21.57	10.75
BREASTW	4.00	3.19	3.05	5.42	3.28
BUPA	32.03	36.26	29.97	31.68	31.19
CENSUS	17.51	24.89	15.47	15.91	16.28
DIABETES	22.99	30.48	23.79	25.96	22.54
IONOSFERA	14.62	15.61	5.56	12.39	16.32
IRIS	2.00	4.00	3.47	6.80	2.47
LANDSAT	16.07	8.95	10.08	18.66	16.74
LETTER	42.83	4.77	7.11	60.72	7.11
PENBASED	12.42	0.66	0.51	18.21	8.72
SEGMENT	8.53	4.66	5.54	8.17	4.92
SHUTTLE	5.61	0.17	0.11	0.53	3.14
SONAR	25.29	18.61	16.01	30.05	25.58
VEHICLE	22.16	35.01	23.33	32.29	20.30
WAVEFORM	13.93	23.33	13.90	26.63	13.25
Promedio	**18.09**	**16.44**	**12.60**	**22.04**	**14.85**

Tabla 3.2. Medidas de complejidad para los diferentes conjuntos de datos

CONJUNTO	F1	F2	F3	Q	QNN	OVER	Qnorm
ABALONE	0.374	0.120	0.588	0.428	0.367	4.460	0.423
BALANCE	0.453	3.000	0.792	0.250	0.110	3.150	0.250
BREASTW	1.353	0.217	0.956	0.035	0.028	0.760	0.035
BUPA	0.165	0.073	0.623	0.438	0.354	6.200	0.400
CENSUS	0.194	0.212	0.726	0.230	0.200	3.917	0.230
DIABETES	0.217	0.252	0.680	0.367	0.275	4.540	0.266
IONOSFERA	0.235	0.043	0.872	0.288	0.134	4.061	0.282
IRIS	2.665	0.054	0.960	0.073	0.040	0.532	0.067
LANDSAT	1.476	0.000	0.910	0.220	0.085	0.970	0.219
LETTER	0.531	2.593	0.961	0.562	0.036	2.723	0.562
PENBASED	1.161	2.060	0.994	0.185	0.006	1.232	0.185
SEGMENT	1.146	0.000	0.967	0.240	0.036	1.535	0.159
SHUTTLE	0.595	0.089	1.000	0.211	0.002	1.941	0.214
SONAR	0.175	0.000	0.827	0.313	0.173	5.363	0.260
VEHICLE	0.505	0.169	0.656	0.626	0.329	5.602	0.547
WAVEFORM	0.516	0.001	0.768	0.207	0.192	2.289	0.195

Si vemos las correlaciones entre los resultados obtenidos en términos del error de clasificación, observamos que los clasificadores LDA y RPART, están altamente correlacionados. Esto significa, que ambos clasificadores presentan un comportamiento similar ante los problemas de clasificación. Debe quedar claro que lo que nos revela esta alta correlación es una tendencia similar, y no que tengan necesariamente los mismos errores de clasificación (ver figura 3.2).

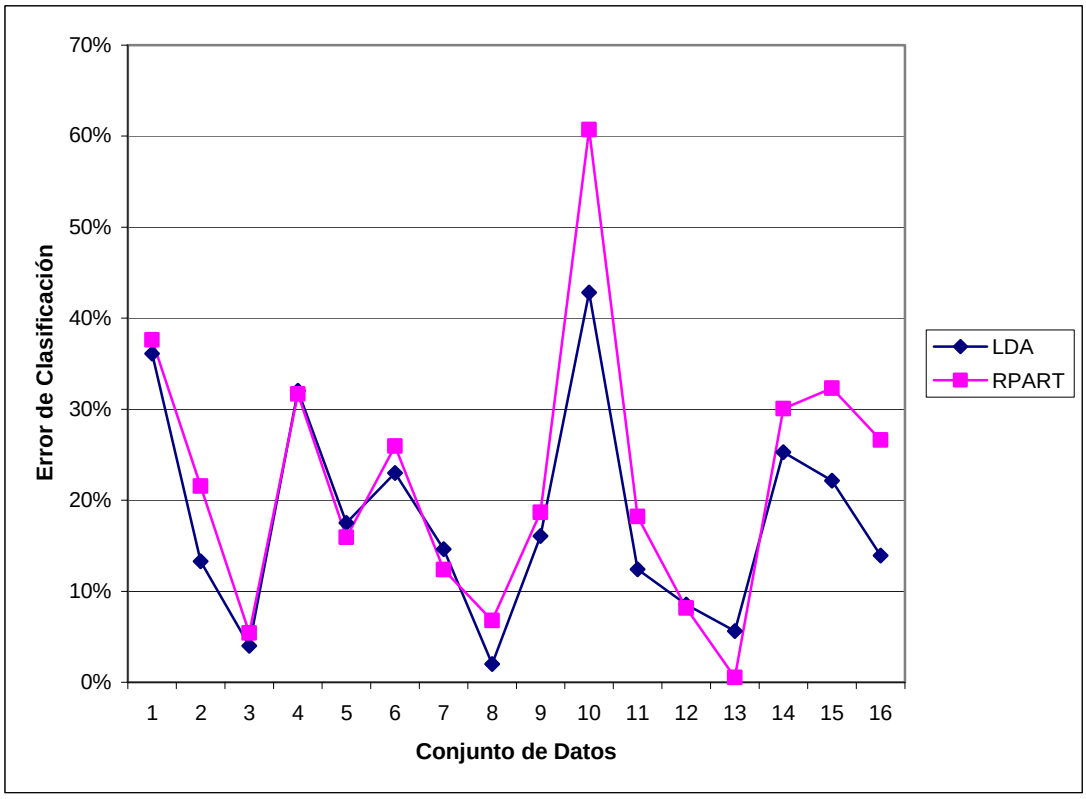

Figura 3.2. Comportamiento de los clasificadores LDA y RPART con diferentes conjuntos de datos

Por otro lado, los otros tres clasificadores: KNN, SVM y LOG, presentan también un comportamiento similar entre sí (y difieren un poco de LDA y RPART). Las correlaciones entre sí para KNN, SVM y LOG, son altas. Estos clasificadores presentan un buen rendimiento en problemas de clasificación cuya frontera de decisión no necesariamente es lineal (ver figura 3.3).

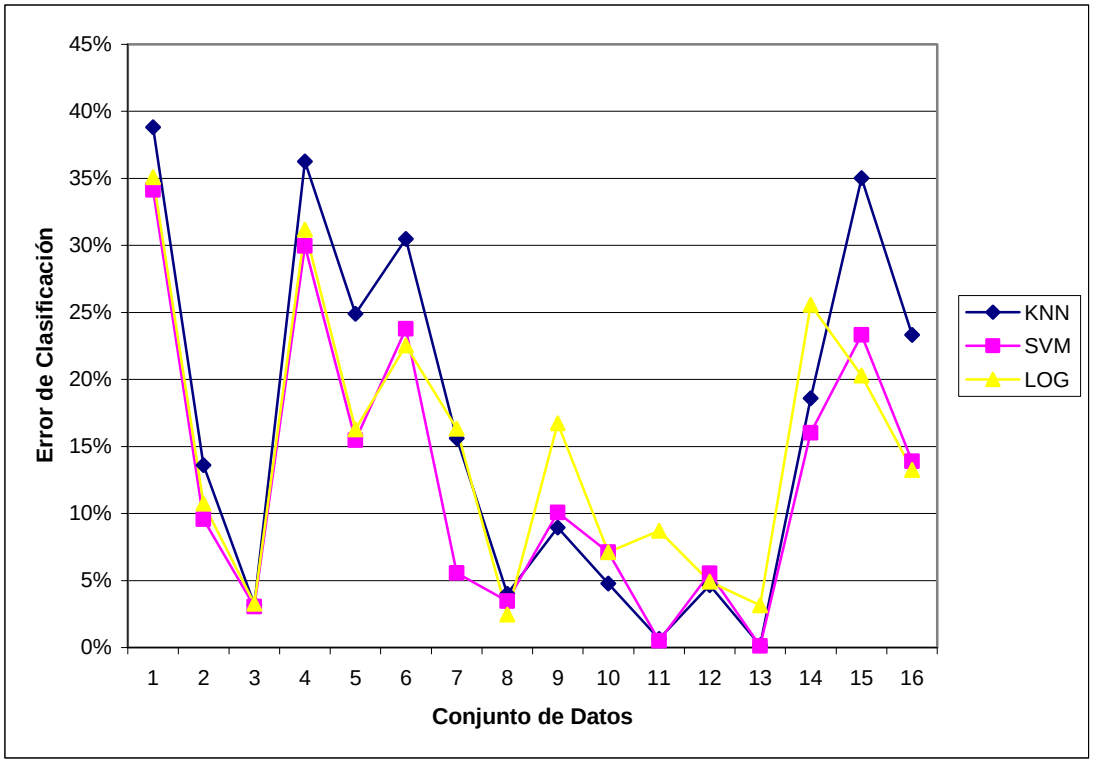

Figura 3.3. Comportamiento de los clasificadores KNN, SVM y LOG Con diferentes conjuntos de datos

Las medidas de complejidad, Q y Qnorm muestran una alta correlación con el clasificador LDA, lo que significa que estas medidas predicen de manera adecuada el comportamiento de un clasificador lineal ante un conjunto de datos. La medida de complejidad, QNN (al igual que la medida F3), presenta una alta correlación con los errores de clasificación de KNN, SVM y LOG, esto revela que esta medida de complejidad está más relacionada a los clasificadores no lineales (ver tabla 3.3 y gráficos 3.4-3.7).

Tabla 3.3. Correlación entre las tasas de error de clasificación y las medidas de complejidad para los 16 conjuntos de datos

	LDA	KNN	SVM	RPART	LOG
LDA	1				
KNN	0.565	1			
SVM	0.673	0.953	1		
RPART	0.930	0.436	0.531	1	
LOG	0.679	0.877	0.918	0.488	1
F1	-0.584	-0.601	-0.527	-0.443	-0.597
F2	0.216	-0.313	-0.272	0.39	-0.278
F3	-0.546	-0.988	-0.953	-0.428	-0.862
Q	0.826	0.585	0.618	0.798	0.555
QNN	0.581	0.992	0.970	0.447	0.900
OVER	0.653	0.836	0.790	0.499	0.819
Qnorm	0.856	0.536	0.570	0.83	0.528

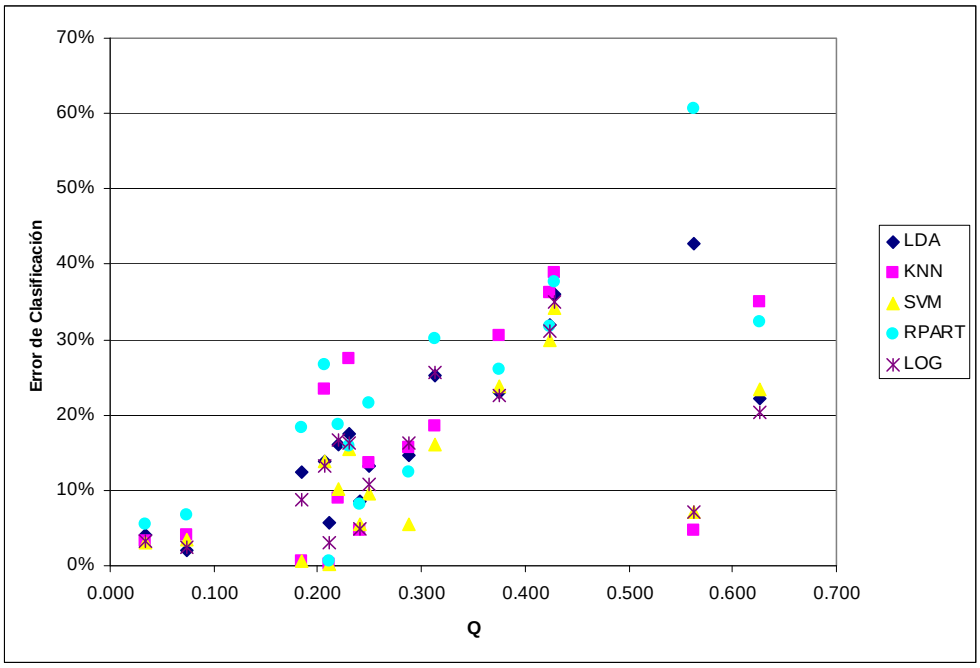

Figura 3.4. Relación entre las tasas de error y la medida de complejidad Q para los cinco clasificadores

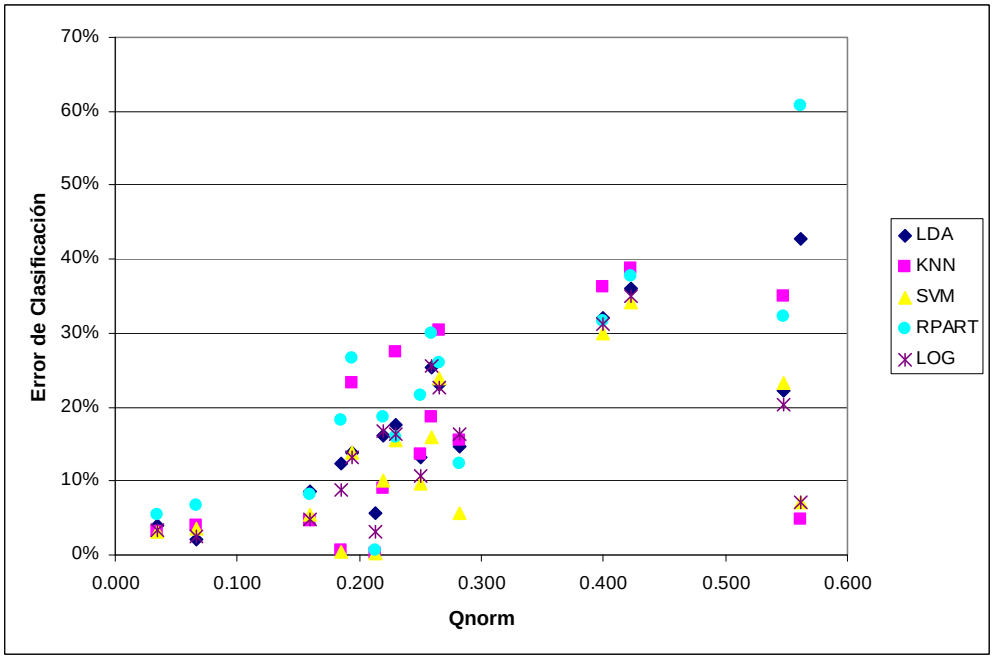

Figura 3.5. Relación entre las tasas de error y la medida de complejidad Qnorm para los cinco clasificadores

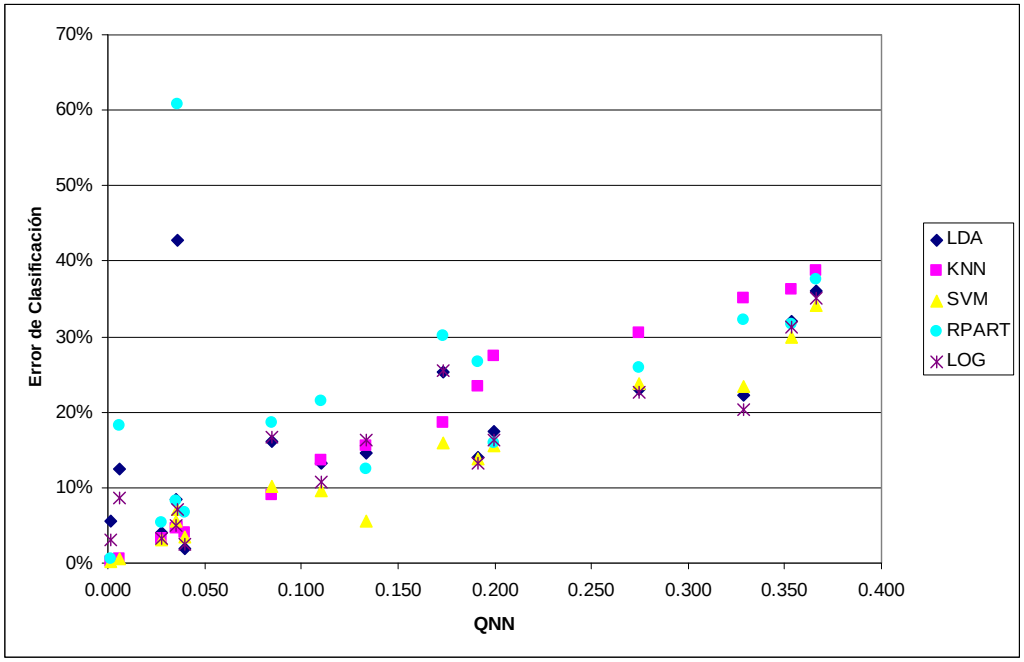

Figura 3.6. Relación entre las tasas de error y la medida de complejidad QNN para los cinco clasificadores

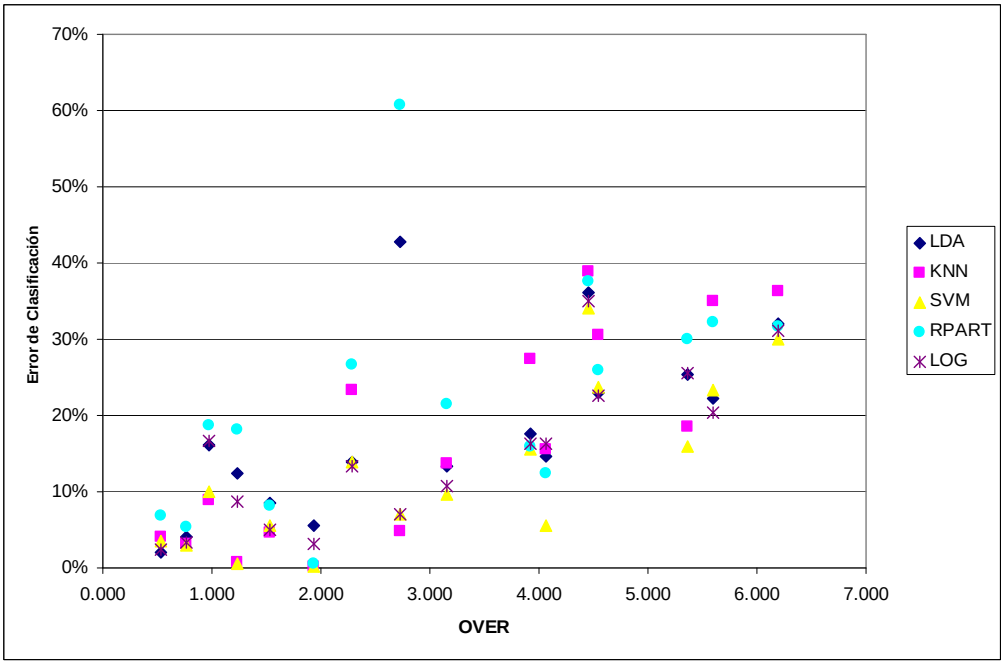

Figura 3.7. Relación entre las tasas de error y la medida de complejidad OVER para los cinco clasificadores

Capítulo IV

Selección de Variables

4.1. Introducción

Actualmente, el crecimiento de los datos se da en ambas dimensiones: filas (número de instancias) y columnas (número de variables). La enorme cantidad de información puede causar serios problemas a los algoritmos de la minería de datos con respecto a la escalabilidad y sus rendimientos. Por ejemplo, en el caso de conjuntos de datos con alta dimensionalidad, la presencia de variables irrelevantes y redundantes degradan el rendimiento de los algoritmos de la minería de datos (Yang y Pederson, 1997; Xing *et al.*, 2001). Por lo tanto, se hace necesario, seleccionar un subconjunto de variables para facilitar y mejorar la labor de la minería de datos.

En teoría, el error de Bayes (o error óptimo), decrece conforme la dimensionalidad de los datos se incrementa. En la práctica, sin embargo, se usa un número fijo de objetos para construir el clasificador y los clasificadores se construyen en base a estos objetos. De esta forma, los estimadores de las densidades de probabilidad, las cuales determinan las fronteras de decisión entre clases, están sesgados por la muestra disponible (Fukunaga, 1990).

Hughes (1968) mostró que la precisión conseguida por un clasificador construido a partir de un número finito de objetos aumenta con la dimensionalidad (p) hasta un cierto punto, a

partir del cual la precisión decrece conforme se incorporan nuevas variables. Esto se conoce como ***el fenómeno de Hughes***, que se ilustra gráficamente en la figura 4.1.

Según Fukunaga y Hayes (1989), cuando el tamaño de la muestra de entrenamiento es relativamente pequeño comparado con la dimensionalidad de los datos, los estimadores por máxima verosimilitud obtenidos son inestables y sesgados, produciendo errores de clasificación más altos. Este fenómeno puede interpretarse de la siguiente manera: Existe un valor óptimo de dimensionalidad que es función del tamaño del conjunto de entrenamiento. Si no se manifiesta este fenómeno, el clasificador construido a partir de un número fijo de muestras se comportará relativamente bien, ya que se aproximará al clasificador de Bayes. Sin embargo, si el número de observaciones en la muestra de entrenamiento es insuficiente y la dimensionalidad de los datos es alta, el fenómeno de *Hughes* se manifiesta. Este fenómeno es más grave cuanto mayor sea la dimensionalidad de los datos.

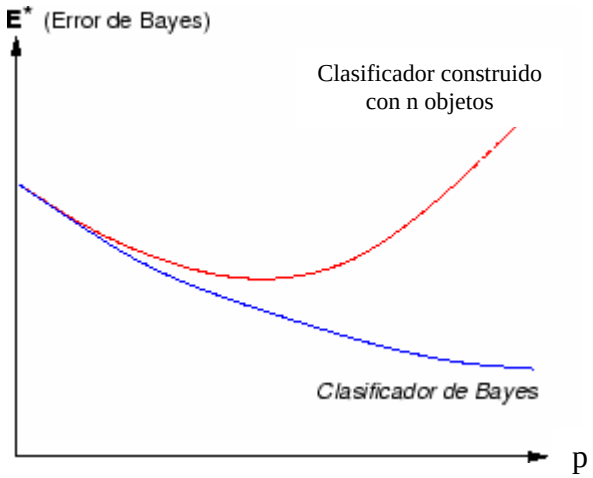

Figura 4.1. Fenómeno de *Hughes*

Como se vio en el capítulo 2, la selección de variables es una técnica usada frecuentemente en el pre-procesamiento de los datos en problemas de alta dimensionalidad, en la minería

de datos. Tradicionalmente la selección de variables se ha enfocado en remover las variables irrelevantes. Sin embargo, en problemas de alta dimensionalidad, remover las variables redundantes, resulta ser igualmente importante (Yu y Liu, 2003).

Varios trabajos en la selección de variables han mostrado ser efectivos para remover las variables irrelevantes y redundantes, incrementando la eficiencia de las tareas de la minería de datos, mejorando el rendimiento en términos de precisión, y haciendo posible que los resultados sean más fáciles de interpretar y entender (Blum y Langley, 1997; Dash y Liu, 1997; Kohavi y John, 1997; Liu y Motoda, 1998).

Queda claro que la reducción de la dimensionalidad a través de la eliminación de las variables irrelevantes y redundantes es necesaria. Sin embargo, el crecimiento de los conjuntos de datos en ambas dimensiones (número de instancias y variables), se presenta como un gran reto para los algoritmos de selección de variables en términos de eficiencia y efectividad. En términos prácticos, la manipulación de conjuntos de datos de muy alta dimensionalidad requiere una gran cantidad de espacio de almacenamiento y tiempo de cálculo, esto significa que los costos computacionales se incrementan severamente y en algunos casos resulta inmanejable. Encontrar un subconjunto óptimo en estas condiciones resulta ser usualmente intratable (Kohavi y John, 1997) y muchos de los problemas relacionados a la selección de variables han mostrado ser NP-hard (Blum y Rivest, 1992).

Recientes trabajos de investigación en la selección de variables están enfocados en enfrentar el reto de tener un gran número de filas (Liu y Motoda, 2002; Liu *et al*., 2003), en conjuntos de datos con alta dimensionalidad (Das, 2001; Xing *et al*., 2001). Muchos de los esfuerzos están enfocados en conseguir algoritmos híbridos, que combinen las ventajas de los algoritmos *filters* y *wrappers*, con el objetivo de mejorar el rendimiento de algún algoritmo de minería de datos en particular. Una excelente referencia para estos algoritmos se encuentra disponible en Liu y Yu (2005). El problema de estas nuevas propuestas está

en que los nuevos algoritmos de selección de variables no reducen la complejidad en tiempo de ejecución de los algoritmos ya existentes.

En este trabajo se propone un algoritmo filtro, el cual establece un *ranking* de las variables, basada en la reducción de las instancias en la región traslapada de las clases. Adicionalmente, proponemos un algoritmo híbrido (*filters* - *wrappers*). El algoritmo filtro propuesto tiene complejidad $O(N)$, mientras que la complejidad del algoritmo híbrido es de $O\left(p \times O\left(clasificador\right)\right)$, donde $O\left(clasificador\right)$ es la complejidad asintótica del clasificador utilizado para evaluar y p es el número de variables. Se presentan resultados comparativos con un algoritmo filtro (RELIEFF) y el algoritmo de envoltura selección hacia delante (SFS, de sus siglas en inglés).

4.2. Métodos de selección de variables

La selección de variables consiste en seleccionar un subconjunto de variables de tal manera que el rendimiento del clasificador mejore, esto es, se reduzca el error de clasificación, sin embargo, en la práctica esto no ocurre siempre.

Dado un conjunto con p variables predictoras, se debe determinar cuál es el mejor subconjunto de q (menor que p) variables, que producen una clasificación más precisa. En términos óptimos lo que se requiere es evaluar el criterio de elección de todas las posibles combinaciones de q variables seleccionadas de las p posibles y, elegir la combinación para la cual tal criterio sea un máximo. Al optar por esta solución surge una dificultad, debido a que el número de posibles subconjuntos es:

$$n_q = \binom{p}{q} = \frac{p!}{(p-q)!q!} \tag{4.1}$$

el cual puede ser un número muy grande, aún para valores moderados de p y q.

De acuerdo con Webb (1999), las estrategias existentes para llevar a cabo la tarea de selección, se pueden categorizar en dos grupos:

1. **Métodos óptimos**: éstos incluyen técnicas de búsqueda intensiva que pueden llevar a una solución óptima global, pero a costa de un gran esfuerzo computacional, lo que implica que se apliquen a problemas de dimensión pequeña. Entre los más usados tenemos a la búsqueda acelerada y métodos de Monte Carlo.
2. **Métodos subóptimos**: con éstos se gana eficiencia computacional a costa de encontrar un subconjunto que no es globalmente óptimo, en contraste al anterior método.

Los métodos subóptimos de selección de variables a su vez se pueden clasificar en tres grupos: Métodos de filtro (*Filters methods*), (Kira y Rendall, 1992; Kohavi y John, 1997; Kim *et al*., 2003), métodos de envoltura (*Wrapper methods*) (Liu y Setiono, 1996; Liu *et al*., 2002; Yu y Liu, 2003) y métodos de selección implícita (Friedman, 1991; Quinlan, 1993).

4.2.1. Métodos filtro

Los métodos filtro para la selección de variables, son técnicas de pre-procesamiento puro. Estos métodos realizan una evaluación de las variables en términos genéricos, sin usar los clasificadores. Mediante los métodos filtro, se evalúa individualmente cada variable y se les asigna un puntaje, para de esta forma establecer un *ranking* de variables y posteriormente seleccionar un subconjunto de variables en base a un valor de corte (umbral), posteriormente a la selección de variables es que se usa el algoritmo de la minería de datos, para generar la regla de clasificación (ver figura 4.2). La principal ventaja de los métodos filtro es su velocidad en términos de procesamiento, ya que tienen por lo general una menor complejidad computacional que los métodos de envoltura (*Wrappers*). Sin embargo, la precisión es típicamente menor (mayor error de clasificación), que los métodos *Wrappers*. A pesar de esta desventaja, la ganancia en velocidad y el de

poder trabajar con conjuntos de datos más grandes, la hacen competitiva con una pérdida mínima de rendimiento. Dentro de los métodos filtro de selección de variables, en esta investigación se va a trabajar con el método *ReliefF*, el cual se describe a continuación.

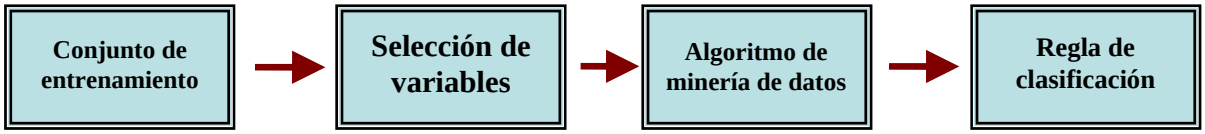

Figura 4.2. Esquema general del método Filtro

Método RELIEF

El método filtro de selección de variables RELIEF, fue introducido por Kira y Rendall (1992). Utiliza la asignación de pesos a las variables para determinar la significancia de cada variable en el contexto de la tarea de clasificación supervisada. La idea clave del método RELIEF es estimar los pesos para las variables de acuerdo con su capacidad para distinguir entre las instancias que están cerca una a la otra. Con este objetivo, dado una instancia, RELIEF busca sus dos vecinos más cercanos: uno de su misma clase (llamada *nearest hit*), y la otra de la clase opuesta (llamada *nearest miss*). El algoritmo original (ver figura 4.3) selecciona aleatoriamente n instancias del conjunto de entrenamiento, donde n es un parámetro definido por el usuario.

La función $diff(Variable, Inst1, Inst2)$, calcula las diferencias entre los valores de la variable para dos instancias. La normalización con *n*, busca garantizar que los pesos de las variables estén en el intervalo $[-1, 1]$.

Los pesos calculados estiman la calidad de las variables. El razonamiento para la fórmula de calcular los pesos, es que una buena variable debe poseer valores similares para las instancias de la misma clase y diferentes valores para instancias en otras clases (clase

opuesta). La complejidad del algoritmo original es $O(n \times N \times p)$, donde n, es el tamaño de la muestra aleatoria, N es número de todas las instancias en el conjunto de entrenamiento y p, es el número de variables.

```
Para todas las variables, asignar W(F) := 0.0
For i =1 to n do
    Begin
       Seleccionar aleatoriamente una instancia I de una clase
       Hallar la instancia más cercana de la misma clases (H)
       Hallar la instancia más cercana  de la clase opuesta (M)

       for F:= 1 to Número de variables do
           W(F) := W(F) – diff(F,I,H)/n + diff(F,I,M)/n
       end
end
```

Figura 4.3. Algoritmo básico de RELIEF

El algoritmo original propuesto por Kira y Rendall (1992), está limitado a solamente dos clases y al uso de un sólo vecino más cercano, al determinar los pesos para las variables. Kononenko (1994) extiende el algoritmo original a más de dos clases y lo denomina RELIEFF. En esta extensión en vez de encontrar un sólo vecino más cercano de la clase opuesta (*near miss*), RELIEFF busca los *k* vecinos más cercanos (*k near misses*), denotado por $M_i(C), i = 1..k$, para cada una de las clases diferentes a *C* y promedia su contribución para actualizar los pesos estimados $W(F)$. El promedio es ponderado por las probabilidades *a priori* de cada clase:

$$W(F) := W(F) - \sum_{i=1}^{k} \frac{diff(F, R, H_i)}{n \times k} + \sum_{C \neq class(R)} \sum_{i=1}^{k} \left[\frac{P(C)}{1 - P(class(R))} \times \frac{diff(F, R, M_i(C))}{n \times k} \right] \quad (4.2)$$

La idea es que el algoritmo estime la habilidad de las variables para separar cada par de clases, sin importar qué par de clases está más cerca. Por otro lado, la normalización utilizando las probabilidades *a priori*, busca que no se exagere la influencia de clases con un número pequeño de instancias.

La complejidad del algoritmo extendido RELIEFF es $O\left(N^2 \times p\right)$, donde N, es el número de todas las instancias en el conjunto de entrenamiento y p es el número de variables.

4.2.2. Métodos de envoltura (*Wrapper*)

Los métodos de envoltura, a diferencia de los métodos filtro requieren de un algoritmo de la minería de datos para realizar la clasificación. Este método cuantifica la calidad de los subconjuntos de variables seleccionadas basándose en como estas variables elegidas clasifican al conjunto de entrenamiento. Debido a esta forma de evaluar los subconjuntos elegidos, es que mediante el método de envoltura se obtienen mejores resultados en términos de precisión, respecto a los modos filtro. Sin embargo, la ganancia de precisión tiene un costo en términos de eficiencia y generalidad, debido a un mayor esfuerzo computacional y que el proceso de selección de variables está asociado a un clasificador específico. La figura 4.4, muestra el esquema general del método de envoltura, notándose claramente la interrelación en la fase de selección de variables del clasificador y el proceso de cuantificar las variables relevantes.

A continuación, se describen dos métodos clásicos, dentro de los métodos de envoltura: Método de selección hacia adelante (Método *Forward*) y el método de selección hacia atrás (Método *Backward*).

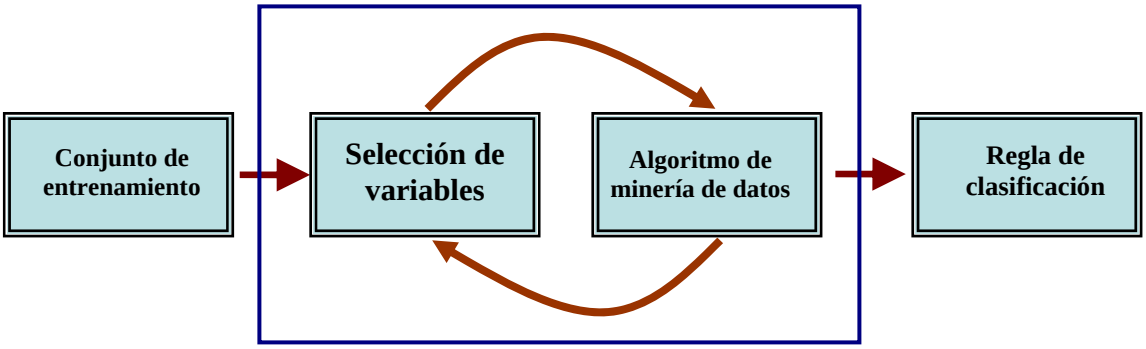

Figura 4.4. Esquema general del método *Wrapper*

Método de selección hacia adelante (Método *Forward*)

La selección de variables hacia adelante, es un procedimiento que consiste en agregar nuevas variables al conjunto de variables ya seleccionadas, una por una, en cada paso, hasta que un criterio de parada se satisfaga.

Sea X_p el conjunto de las p variables en el conjunto de datos y Y_q, el conjunto de las q variables seleccionadas. En términos generales, el algoritmo puede plantearse como sigue:

1. El algoritmo se inicia con un conjunto vacío de variables, esto es, $Y_0 = \phi$.
2. Supongamos que después de k pasos se han seleccionado k variables y se tiene el conjunto Y_k con las k variables seleccionadas.
3. Para cada una de las variables ξ_j no seleccionadas aún (es decir, en $X_p - Y_k$), se calcula la tasa T, de buena clasificación de esta variable unida a las ya seleccionadas, o sea, $T_j = T\left(Y_k \cup \xi_j\right)$. La variable a elegir es aquella que produzca el máximo valor de T_j.
4. El proceso termina cuando cada una de las variables $\xi_j = X_p - Y_k$, se genera una tasa de clasificación menor, o cuando se exceda de un número especificado previamente de características permitidas.

La principal desventaja de este método es que no tiene un mecanismo para eliminar variables que ya han sido seleccionadas. Esto constituye una desventaja ya que existe la posibilidad de que al incluir una nueva variable, algunas de las seleccionadas anteriormente sean irrelevantes.

La complejidad del método de selección hacia adelante (Método Forward) es $O\left(p \times O\left(clasificador\right)\right)$, donde $O\left(clasificador\right)$, es la complejidad del clasificador que se usa para evaluar a las variables y p, es el número de variables.

Método de selección hacia atrás *(Método* Backward*)*

La selección de variables hacia atrás, es un procedimiento que consiste en eliminar variables del conjunto de variables ya existentes, una por una, en cada paso, hasta que el criterio de parada se cumpla.

Sea X_p el conjunto de las p variables en el conjunto de datos y $\tilde{\mathrm{Y}}_k$, el conjunto resultante luego eliminar k variables. En términos generales, el algoritmo puede plantearse como sigue:

1. El algoritmo se inicia con un conjunto conteniendo todas las variables, esto es, $\tilde{\mathrm{Y}}_0 = \mathrm{X}_p$.
2. Supongamos que después de k pasos se han eliminado k variables y se tiene el conjunto $\tilde{\mathrm{Y}}_k$ que no incluye las k variables eliminadas.
3. Se procede a calcular la tasa de buena clasificación T_j excluyendo cada una de las variables ξ_j no eliminadas aún (es decir, en $\tilde{\mathrm{Y}}_k$), esto es, $\mathrm{T}_\mathrm{j} = \mathrm{T}\left(\tilde{\mathrm{Y}}_\mathrm{k} - \xi_\mathrm{j}\right)$. La variable a eliminar es aquella que produzca el máximo valor de T_j.

4. El proceso termina cuando se eliminan cada una de las variables que quedan en el modelo y se generan tasas de clasificación menores, o cuando se exceda de un número especificado previamente de características a eliminar.

En cada paso de este método, la variable que se descarta se determina considerando la dependencia estadística entre las variables retenidas. Cuando se descarta una variable no hay forma de recuperarla si en un paso posterior se comprobara que ésta resulta necesaria. Este algoritmo es computacionalmente más demandante que el método de selección hacia adelante, ya que en los primeros pasos se evalúa la función discriminante en espacios de mayor dimensionalidad que el método de selección hacia adelante. Proporciona, generalmente, mejores resultados, en términos de reducir el error de clasificación, que el método de selección hacia adelante, ya que las últimas evaluaciones de este método, se hacen en espacios de menor dimensionalidad y se comparan con el resultado de considerar todas las variables conjuntamente.

La complejidad del método de selección hacia atrás (Método **Backward**), es $O\left(p \times O\left(clasificador\right)\right)$, donde $O\left(clasificador\right)$, es la complejidad del clasificador que se usa para evaluar a las variables y p, es el número de variables.

4.2.3. Métodos de selección implícita

En el caso de los métodos de selección implícita, la selección de variables es incluida dentro de la fase de aprendizaje a partir del conjunto de entrenamiento (ver figura 4.5). Durante este proceso, el algoritmo remueve las variables redundantes del conjunto de datos y/o simplifica al conjunto de entrenamiento. Un ejemplo clásico de este tipo de selección es el algoritmo C4.5 que fue creado por Quinlan (1993). El algoritmo C4.5 induce el árbol de decisión incluyendo sólo las variables necesarias, de esta forma las variables innecesarias no son incluidas en el árbol de clasificación resultante. Otro ejemplo, de este tipo de selección de variables, es MARS (Friedman, 1991)

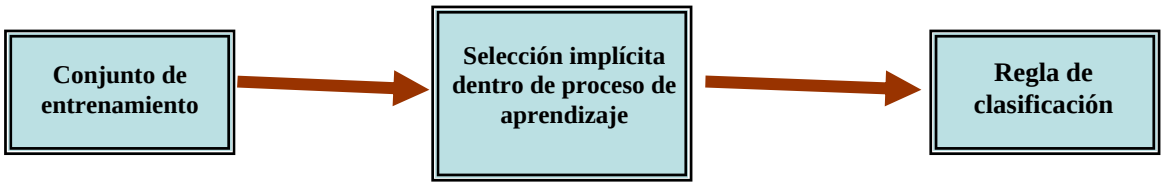

Figura 4.5. Esquema general del método de selección implícita

4.3. Métodos propuestos de selección de variables

Como se ha mencionado anteriormente, en esta investigación se proponen dos métodos de selección de variables. Un método filtro, que llamaremos Wfeat y un método híbrido que combina el método filtro anterior con el método de envoltura, selección hacia delante (SFS, de sus siglas en inglés). Ambos métodos propuestos, buscan mejorar la eficiencia de los algoritmos clásicos, manteniendo (o mejorando) la precisión de los mismos.

4.3.1. Método basado en la reducción de la complejidad del problema de clasificación (Wfeat)

Este método filtro de selección de variables, busca identificar las variables irrelevantes en un problema de clasificación supervisada. Con este objetivo, se evalúa la calidad de las variables, asignando a cada una de ellas un peso, de acuerdo con su capacidad para reducir la complejidad de la frontera de decisión.

Para usar las medidas de calidad de las instancias en determinar la complejidad de un conjunto de datos procedemos como sigue:

1. Para cada instancia del conjunto de entrenamiento, se calculan p medidas de calidad univariado $Q\left(F_j\right), \quad j=1, 2, \cdots p$. Donde F_j es la j-ésima variable.
2. Para cada variable se calcula el promedio de la medida calidad de todas las instancias en el problema.

$$Q\left(F_j\right)_{\text{Prom.}} = \frac{\sum_{i=1}^{N} Q_i}{N} \tag{4.3}$$

3. A cada variable le corresponde un peso de:

$$W\left(F_j\right) = \exp\left(Q\left(F_j\right)_{prom} - 1 \right)$$

4. Con los pesos obtenidos por las variables se establece un *ranking*, que va de las variables más relevantes a las variables más irrelevantes.

Naturalmente, el siguiente paso es determinar con cuántas variables nos quedamos. Una alternativa es quedarse con un porcentaje del número de variables originales (por ejemplo, el 60%) y la otra alternativa es usar un gráfico qué realice un *plot* de los pesos de las variables y determinar de forma gráfica que variables son más relevantes.

Complejidad computacional. En el paso 1 se calcula la medida de calidad Q para cada variable, lo cual se realiza en tiempo lineal al número de instancias N y variables p (ver sección 3.5). Por otro lado, los pasos 2 y 3 se hacen en N pasos. Finalmente, el paso 4 del algoritmo tiene una complejidad del orden $p \log p$. Por consiguiente, la complejidad asintótica del método filtro Wfeat es $O(Np)$, donde N, es el número de todas las instancias en el conjunto de entrenamiento y p el número de variables. Este método es eficiente en términos del tiempo de computación y los resultados experimentales mostrarán que también es competitivamente preciso con los métodos filtro existentes.

La desventaja de este método tiene que ver con su naturaleza, esto es, la mayoría de los métodos filtro no son capaces de identificar y remover las variables redundantes. Este es el motivo por el que el número de variables es relativamente alto comparado con los métodos de envoltura. En la siguiente sección presentamos un eficiente algoritmo, que trata de subsanar este problema al eliminar las variables redundantes y quedarse con las más relevantes.

4.3.2. Método híbrido de selección de variables (WfeatSFS)

En la sección anterior presentamos un eficiente método filtro y competitivamente preciso para la selección de variables. El problema con este método, es que retiene algunas variables redundantes. Aquí presentamos el algoritmo híbrido WfeatSFS.

Sea $\mathrm{X}_p = \{F_1, F_2, F_3, \cdots F_p\}$ el conjunto de las p variables en el conjunto de datos y Y_q, el conjunto de las variables seleccionadas hasta el paso q. En términos generales, el algoritmo puede plantearse como sigue:

1. El algoritmo inicia con la aplicación de la selección de variables Wfeat, la cual genera una secuencia ordenada de variables seleccionadas por Wfeat, en términos de su relevancia: $\{\tilde{F}_1, \tilde{F}_2, \tilde{F}_3, \cdots, \tilde{F}_d\},\ d \le p$.
2. Se selecciona la primera variable en importancia, generada por el método Wfeat. El algoritmo se inicia con un conjunto incluyendo esta variable, esto es, $\mathrm{Y}_1 = \{\tilde{F}_1\}$. Se calcula la tasa de buena clasificación con esta única variable: $T_{\max} = T(\mathrm{Y}_1)$ y $j = 1$
3. Para j = 2, 3, ..., d. Hacer:
4. Se calcula la tasa T, de buena clasificación del conjunto de variables en Y, unida a la siguiente variable respetando el ranking inicial. Esto es, en el j-ésimo paso, se calcula: $T_j = T(\mathrm{Y}_{j-1} \cup \tilde{F}_j)$. La variable $\tilde{F}_j$ ingresa al conjunto de variables seleccionadas Y , si $T_j > T_{\max}$, $T_{\max} = T_j$ y $j = j + 1$
5. Si $j < d$, ir al paso 3.
6. El proceso termina cuando cada una de las variables seleccionadas por Wfeat, ha sido probada en forma ordenada si es seleccionada usando el método filtro. En el conjunto final de variables seleccionadas no se incluirán aquellas variables que no aporten una mejora en la tasa de buena clasificación (variables redundantes).

La justificación de que este método híbrido funciona, es que si se tienen dos variables redundantes, estas tendrán aproximadamente el mismo peso (W), de forma tal que el método filtro las ordenará en posiciones contiguas. Por consiguiente, si se selecciona una variable de la secuencia ordenada generada por Wfeat, y la siguiente es redundante con respecto a la seleccionada, su aporte adicional a la tasa de clasificación será nulo o negativo.

Complejidad computacional. La complejidad en tiempo del método de selección de variables WfeatSFS en el peor de los casos es $O\left(p \times O\left(clasificador\right)\right)$, donde $O\left(clasificador\right)$ es la complejidad asintótica del clasificador utilizado para hallar la tasa de buena clasificación en los pasos 2 y 4; y p es el número de todas las variables.

En la siguiente sección se presentan resultados experimentales, en los que se comparan los dos métodos propuestos, usando 12 conjuntos de datos con el algoritmo filtro RELIEFF y el algoritmo de envoltura selección hacia adelante SFS).

4.4. Evaluaciones experimentales

4.4.1. Metodología

Se eligieron 12 conjuntos de datos de la base de datos de la UCI (*Abalone*, *Breastw*, *Bupa*, *Census*, *Diabetes*, *Ionosfera*, *LandSat*, *Penbased*, *Segment*, *Shuttle*, *Sonar y Waveform*), para evaluar el rendimiento de los dos métodos propuestos (Wfeat y WfeatSFS) y compararlos con el método filtro RELIEFF y el método de envoltura de selección de variables hacia delante (SFS). En el caso de los métodos filtro (ReliefF y Wfeat), se procedió a considerar las primeras 60% de las variables ordenadas en cuanto a su relevancia como las variables seleccionadas por estos métodos.

A cada conjunto de datos se les aplicó cada uno de los cuatro métodos de selección de variables y, con las variables seleccionadas, se procedió a determinar el error de clasificación usando validación cruzada con los clasificadores: LDA, KNN y RPART. Para comparar la eficiencia de los métodos en este estudio, se procedió a computar los costos computacionales en segundos (CCS), trabajando con cada uno de los métodos de selección bajo las mismas condiciones.

4.4.2. Resultados experimentales

La tabla 4.1. muestra las tasas relativas promedio para los ocho conjuntos de datos, de los errores de clasificación usando las variables seleccionadas con respecto al error de clasificación usando todas las variables.

Observamos que para el algoritmo RELIEFF, se tiene un incremento en los errores de clasificación promedio del orden del 17% para el clasificador LDA, 6% para el clasificador KNN y del 11% para el clasificador RPART. El método de envoltura SFS, muestra un aumento en el error clasificación del 1% para el clasificador LDA, mientras que para el clasificador KNN, el error de clasificación promedio disminuye en un 5% y 6% para el RPART. El método filtro propuesto Wfeat, produce un incremento promedio en los errores de clasificación del 7% para LDA y 5% para RPART, y una disminución del 6% para el clasificador KNN. Finalmente, el método híbrido propuesto WfeatSFS, tiene un 4% por encima del error de clasificación promedio usando todas las variables, para el clasificador LDA; un 10% por debajo para el clasificador KNN y un 2% por debajo para el clasificador RPART.

Tabla 4.1. Tasas relativas promedio de los errores de clasificación usando las variables seleccionadas para los ocho conjuntos de datos

	ReliefF	Wrapper (SFS)	Wfeat	WfeatSFS
LDA	1.17	1.01	1.07	1.04
RPART	1.11	0.94	1.05	0.98
KNN	1.06	0.95	0.94	0.90
Promedio	1.11	0.97	1.02	0.97

Si comparamos el promedio de las tasas relativas entre el método ReliefF y el método Wfeat, se observa que en términos del error de clasificación se tiene un mejor rendimiento con el método que se ha propuesto. En cuanto a los métodos SFS y WfeatSFS, se observa en promedio un empate en términos de rendimiento.

Las tablas 4.2, 4.4 y 4.6, muestran los errores de clasificación para cada uno de los doce conjuntos de datos, los cuatro métodos de selección de variables y los tres clasificadores. Mientras que las tablas 4.3, 4.5 y 4.7 detallan los costos computacionales medidos en segundos.

Comparando los tiempos de computación, observamos que el método filtro Wfeat es bastante rápido, mucho más eficiente que el método ReliefF. Por otro lado, el método híbrido WfeatSFS, también se muestra más rápido que su contraparte SFS.

Tabla 4.2. Error de clasificación con las variables seleccionadas por los diferentes métodos usando el clasificador LDA

Conjunto	Todas	Relief	Wrapper (SFS)	Wfeat	WfeatSFS
Abalone	36.12	38.58	35.60	38.18	35.48
Breastw	4.00	4.80	3.69	4.83	4.16
Bupa	32.03	33.39	33.22	32.41	37.41
Census	17.51	18.67	17.45	17.57	17.46
Diabetes	22.99	32.84	23.46	23.13	23.23
Ionosfera	14.62	16.98	16.70	17.26	16.24
Landsat	16.07	16.41	16.42	17.15	17.61
Penbased	12.42	18.11	12.42	18.29	16.50
Segment	8.53	19.62	8.40	9.14	8.54
Shuttle	5.61	5.66	4.31	5.49	5.00
Sonar	25.29	26.63	25.38	25.96	22.60
Waveform	13.93	13.66	13.90	13.86	13.66
Promedio	**17.43**	**20.45**	**17.58**	**18.61**	**18.16**

Tabla 4.3. Tiempos de computación en segundos para los diferentes métodos usando el clasificador LDA

Conjunto	ReliefF	Wrapper (SFS)	Wfeat	WfeatSFS
Abalone	393.4	119.1	1.8	35.7
Breastw	12.3	51.7	0.5	17.1
Bupa	1.8	23.3	0.1	8.5
Census	3818.7	1579.9	17.4	394.8
Diabetes	10.1	44.2	0.5	13.9
Ionosfera	1.8	129.5	0.5	32.8
Landsat	1107.6	2435.1	32.8	186.0
Penbased	3366.1	1900.2	21.7	153.6
Segment	180.2	378.1	5.9	40.8
Shuttle	11284.6	1501.6	60.9	483.5
Sonar	0.7	357.1	0.9	60.2
Waveform	894.2	2973.8	19.0	286.8
Promedio	**1756.0**	**957.8**	**13.5**	**142.8**

Tabla 4.4. Error de clasificación con las variables seleccionadas por los diferentes métodos usando el clasificador KNN

Conjunto	Todas	Relief	Wrapper (SFS)	Wfeat	WfeatSFS
Abalone	38.82	42.03	39.96	42.40	40.31
Breastw	3.19	4.10	3.09	3.34	3.02
Bupa	36.26	38.67	45.10	39.48	39.36
Census	24.89	19.89	22.81	15.78	17.24
Diabetes	30.48	39.30	26.64	28.49	26.98
Ionosfera	15.61	14.47	6.82	12.36	7.69
Landsat	8.95	9.74	9.40	9.63	9.95
Penbased	0.66	2.19	0.66	1.84	1.65
Segment	4.66	9.54	3.12	5.50	3.28
Shuttle	0.17	0.21	0.05	0.15	0.07
Sonar	18.61	18.65	18.56	15.96	17.40
Waveform	23.33	18.20	18.46	19.21	17.62
Promedio	**17.14**	**18.08**	**16.22**	**16.18**	**15.38**

Tabla 4.5. Tiempos de computación en segundos para los diferentes métodos usando el clasificador KNN

Conjunto	Relief	Wrapper (SFS)	Wfeat	WfeatSFS
Abalone	393.4	70.2	1.8	34.5
Breastw	12.3	39.1	0.5	11.0
Bupa	1.8	5.4	0.1	2.9
Census	3818.7	2450.6	17.4	1150.0
Diabetes	10.1	17.8	0.5	6.2
Ionosfera	1.8	61.2	0.5	13.9
Landsat	1107.6	5032.6	32.8	375.2
Penbased	3366.1	600.7	21.7	57.9
Segment	180.2	180.3	5.9	27.8
Shuttle	11284.6	4700.3	60.9	1851.0
Sonar	0.7	122.7	0.9	24.8
Waveform	894.2	5582.8	19.0	360.4
Promedio	**1755.9**	**1572.0**	**13.5**	**326.3**

Tabla 4.6. Error de clasificación con las variables seleccionadas por los diferentes métodos usando el clasificador RPART

Conjunto	Todas	Relief	Wrapper (SFS)	Wfeat	WfeatSFS
Abalone	37.62	39.56	37.89	39.69	37.49
Breastw	5.42	4.80	4.56	5.07	4.56
Bupa	31.68	35.01	32.52	35.88	32.75
Census	15.91	18.51	15.43	15.93	15.43
Diabetes	25.96	34.27	25.26	26.93	25.73
Ionosfera	12.39	11.40	10.20	11.62	9.86
Landsat	18.66	19.02	18.39	19.06	18.74
Penbased	18.21	19.99	16.81	20.10	19.61
Segment	5.54	14.14	8.02	8.37	8.37
Shuttle	0.53	0.53	0.53	0.53	0.53
Sonar	30.05	29.42	19.62	29.23	25.10
Waveform	26.63	26.92	26.00	26.84	26.38
Promedio	**19.05**	**21.13**	**17.94**	**19.94**	**18.71**

Tabla 4.7. Tiempos de computación en segundos para los diferentes métodos usando el clasificador RPART

Conjunto	Relief	Wrapper (SFS)	Wfeat	WfeatSFS
Abalone	393.4	672.0	1.8	230.0
Breastw	12.3	96.6	0.5	32.7
Bupa	1.8	50.1	0.1	20.2
Census	3818.7	14217.0	17.4	4436.6
Diabetes	10.1	99.3	0.5	44.1
Ionosfera	1.8	561.1	0.5	88.8
Landsat	1107.6	5420.2	32.8	880.4
Penbased	3366.1	9787.9	21.7	1009.5
Segment	180.2	961.0	5.9	210.5
Shuttle	11284.6	9935.3	60.9	2939.5
Sonar	0.7	897.6	0.9	144.5
Waveform	894.2	8636.2	19.0	1632.2
Promedio	**1756.0**	**4277.9**	**13.5**	**972.4**

Capítulo V

Detección de Ruido en Clasificación Supervisada y su Efecto sobre el Error de Clasificación

5.1. Introducción

La situación ideal para extraer conocimiento de una base de datos es que no tenga datos anormales y/o ruido. Sin embargo, las bases de datos reales no están necesariamente "limpias" por lo que requieren un trabajo de limpieza previo antes realizar la tarea de la minería de datos. El ruido en un conjunto de datos puede reducir el rendimiento del algoritmo incrementando el error de clasificación, afectar el tiempo requerido para construir el clasificador y generar clasificadores o reglas de decisión mucho más complejas.

Dado un conjunto de datos, su calidad puede estar usualmente caracterizada por dos fuentes de información: las variables predictoras y la variable categórica que determina la clase de pertenencia de cada instancia. La calidad en las variables está determinada por su capacidad para representar a las instancias que van a ser clasificadas; y la calidad en la variable de clases se determina en la medida de que cada instancia esté correctamente asignada a una clase.

La calidad del conjunto de datos está determinada por dos factores, internos y externos: El factor interno revela si las variables y las clases han sido bien seleccionadas y definidas, el factor externo indica errores introducidos en las variables o en la asignación de las clases

(sistemáticamente o artificialmente). Específicamente, cuando una instancia causa problemas debido a una razón externa, se dice que la instancia contiene ruido. En Quinlan (1986) se asumen como ruido los errores no sistemáticos en los valores de las variables así como en la información de las clases.

De acuerdo con Zhu *et al*. (2003), el error de clasificación mínimo está condicionado por la calidad de la información contenida en el conjunto de entrenamiento y el sesgo inductivo del algoritmo que se utiliza para realizar la clasificación. Esto significa, que mejorando la calidad de la información del conjunto de entrenamiento se reducirá el error de clasificación, debido a que el rendimiento del algoritmo clasificador mejora sustancialmente cuando se limpia el conjunto de entrenamiento.

Según Zhu *et al*. (2003, 2006), podemos distinguir dos tipos de ruido:

a) **Ruido en las variables**: Son los errores que se presentan al ingresar los valores de las variables. Las fuentes del ruido en las variables pueden ser:
   1. Errores en los valores de las variables.
   2. Variables con valores perdidos o desconocidos.
   3. Variables incompletas
   4. Datos redundantes

b) **Ruido en las clases**: Son errores introducidos al asignar las clases a las instancias. La presencia de ruido en las clases puede deberse a la subjetividad, errores al ingresar los datos o información errada para asignar una clase a la instancia. Se tienen dos posibles fuentes de ruido en las clases:
   1. Instancias contradictorias. Instancias con los mismos valores para todas las variables consideradas aparecen asignadas a dos o más clases diferentes.
   2. Error en la clasificación. Instancias asignadas a una clase de manera incorrecta. Este tipo de error es común cuando se tienen clases con valores similares para las variables.

La existencia de ruido en las clases afecta de manera significativa el rendimiento de los algoritmos de clasificación existentes, ya que modifica las fronteras de las clases y se hace más complejo determinar la frontera de decisión. Las instancias que son ruido tienden a clasificar incorrectamente a instancias correctamente etiquetadas, esto tiene un efecto directo sobre la precisión de los algoritmos de clasificación.

En este trabajo se propone una estrategia de búsqueda e identificación de instancias ruidosas en el conjunto de entrenamiento. Se realiza un estudio comparativo del efecto de la eliminación del ruido sobre el error de clasificación. La técnica propuesta identifica eficientemente las instancias ruidosas, las cuales son posteriormente eliminadas del conjunto de entrenamiento. Con las instancias restantes se construye un modelo de clasificación, el cual es validado usando validación cruzada. El método de eliminación de ruido es aplicado a varios conjuntos de datos y se verifican los efectos sobre el error de clasificación usando tres clasificadores: Análisis discriminante lineal (LDA, por sus siglas en inglés), particionamiento recursivo (RPART) y k-vecinos más cercanos (KNN).

5.2. Trabajos previos

El problema de clasificación en presencia de ruido ha llamado la atención de varios investigadores del área de aprendizaje automático y algunos algoritmos de aprendizaje inductivo tienen mecanismos para tratar el ruido. Quinlan (1986) demostró que cuando se tienen altos niveles de ruido en las clases, realizar una limpieza en el conjunto de entrenamiento antes de construir el clasificador, mejora el poder predictivo del algoritmo. Por ejemplo, el proceso de poda en los árboles de decisión está destinada a reducir el efecto de sobre entrenamiento de los árboles debido a la presencia de ruido en el conjunto de entrenamiento. Otra alternativa es el uso de la combinación de clasificadores *Bagging* y *Boosting* (Breiman, 1996, 1998, 1999). Lamentablemente los métodos de poda y la combinación de clasificadores resuelven parcialmente el problema, aún así, la presencia de ruido puede afectar dramáticamente el error de clasificación, especialmente cuando el nivel de ruido es relativamente alto. Gamberger *et al*. (1999, 2000), sugieren que el ruido debe ser “eliminado” antes de la construcción del clasificador, para que de esta forma las

instancias ruidosas no influyan en la elaboración del clasificador. De acuerdo con esto, la solución al problema de ruido consiste en limpiar los datos de alguna forma, para mejorar la calidad de los conjuntos de datos existentes.

Existen diversos criterios propuestos para identificar instancias incorrectamente etiquetadas. Por ejemplo, Guyon *et al.* (1996) remueven las instancias ruidosas utilizando el criterio de la mayor ganancia de información. Mientras que Gamberger *et al*. (1999), usan el método de *saturation filter* para determinar la reducción del valor CLCH (*Complexity of the Least Complex correct Hypothesis*).

En otros métodos, los datos considerados potencialmente como ruido son detectados y eliminados usando C4.5 (John, 1995, Zhu *et al*., 2003), o por redes neuronales (Zeng y Martinez, 2003). Algunos autores (John, 1995, Brodley y Friedl 1996, 1999), utilizan un esquema similar al de remover valores anormales en regresión, esto es, se utiliza el mismo modelo para detectar los *outliers* y determinar el modelo final de los datos una vez que los *outliers* han sido eliminados. John (1995), presenta un clasificador robusto basado en árboles de decisión. El estudio consiste en eliminar los valores anormales en el conjunto de entrenamiento mediante el algoritmo ROBUSTC45 (ver figura 5.1), el cual ejecuta repetidamente el clasificador C-4.5 y procede a eliminar las instancias que son mal clasificadas en el conjunto de entrenamiento, hasta que todas las instancias que quedan en el conjunto de entrenamiento reducido sean correctamente clasificadas. En un estudio más reciente realizado por Acuña y Rodríguez (2004), se encontró que los clasificadores se ven poco afectados al remover los valores anormales, dicho efecto es relativo y su impacto depende del clasificador y del conjunto de datos.

```
ROBUST45(TrainingData)
    Repeat {
        T <- C45BuildTree(TrainingData)
        T <- C45PruneTree(T)
            For each record in TrainingData
                If  T misclassifies record then
                    Remove record from TrainingData
    } until T correctly classifies all instances in TrainingData
```

Figura 5.1. Algoritmo ROBUST45

Brodley y Friedl (1996,1999), desarrollan una estrategia similar a la de John (1995), pero utilizan la validación cruzada con una sola iteración, en vez de ejecutar múltiples iteraciones. Ellos proponen un modelo para filtrar las instancias como se muestra en la figura 5.2. El conjunto de entrenamiento inicial que contiene ruido es filtrado utilizando varias estrategias, para obtener un nuevo subconjunto limpio, con el cual se construyen los clasificadores. Basándose en este modelo, utilizan varios algoritmos de clasificación para construir un filtro utilizando el criterio de que una instancia es considerada como ruido si es mal clasificada por uno o más clasificadores cuando pertenece al conjunto de prueba, usando validación cruzada-N. Específicamente, el método consiste en los siguientes pasos:

1. Elegir M algoritmos de clasificación.
2. El conjunto de entrenamiento E es dividido en N subconjuntos disjuntos (usando validación cruzada-N).
3. Con cada uno de los M algoritmos, se construyen los clasificadores utilizando N-1 subconjuntos.
4. Se utilizan los M algoritmos de clasificación para asignar las instancias a una clase, de esta forma, cada instancia recibe M etiquetas de pertenencia a una clase.
5. El filtro compara la clase original de cada instancia con las M asignaciones de los clasificadores y decide si debe o no debe remover la instancia usando diferentes criterios tales como el consenso o la decisión por mayoría.

6. El conjunto de entrenamiento filtrado se utiliza para construir el clasificador de nuevas instancias.

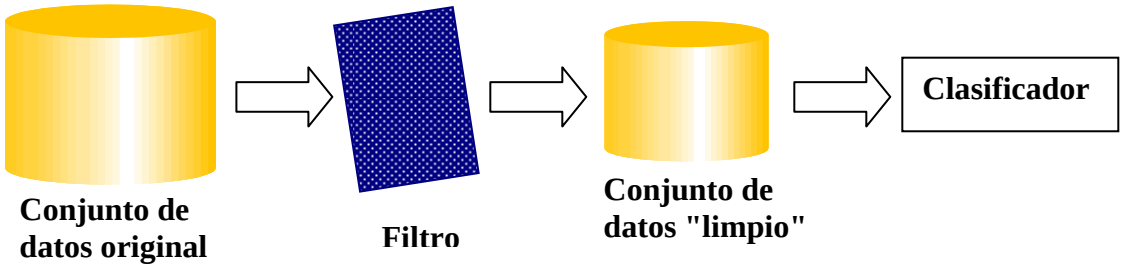

Figura 5.2. Modelo para la identificación y eliminación de ruido en las clases

Existen diversas variaciones al método, dependiendo del criterio que se utiliza para determinar si una instancia es ruido en el paso 5, y del número de clasificadores que se utilizan como filtro. En Brodley y Friedl (1999), se evalúan empíricamente diferentes configuraciones utilizando un solo clasificador como filtro (M=1); o varios clasificadores (M>1) con variaciones en el criterio de voto por consenso o por mayoría. El criterio por consenso significa que una instancia será considerada ruido si es mal clasificada por todos los clasificadores que actúan como filtro. Para determinar que una instancia es ruido por mayoría se requiere que más de la mitad de los clasificadores lo clasifiquen incorrectamente. En caso de que se use un solo clasificador como filtro, una instancia mal clasificada será considerada ruido y debe ser removida.

Zhu *et al*. (2003, 2006) se enfocan en el problema de limpiar conjuntos de datos grandes o bases de datos distribuidas. En su esquema, ellos realizaron en primer lugar una partición del conjunto de entrenamiento completo (*E*) en subconjuntos disjuntos, cada uno de los cuales es lo suficientemente pequeño para ser procesado por un algoritmo de la clasificación. Con cada subconjunto se generan clasificadores (reglas de decisión), las cuales son utilizados para evaluar el conjunto completo. Para una instancia en particular I, se contabiliza como error al número de veces que ha sido identificado como ruido por

todos los subconjuntos. Una instancia con valores más altos en el conteo del error tendrá una probabilidad más alta de ser considerada ruido. Para identificar y eliminar las instancias ruidosas ellos utilizan dos esquemas (mayoría y *non-objection*).

Entre otras soluciones, se tiene al algoritmo introducido por Lawrence y Schölkopf (2001), quienes presentan fundamentos teóricos e incluyen elegantemente el ruido en las clases en el modelo de clasificación. Li (2004) propone tres soluciones, basadas en las ideas presentadas por Lawrence y Schölkopf (2001). Los algoritmos propuestos por ellos tratan de que el clasificador estándar tolere la presencia de ruido.

Como podemos observar existen varios trabajos para la detección de ruido en problemas de clasificación supervisada, pero no ocurre lo mismo cuando se trata de problemas de clasificación no supervisada (*Clustering*). Un reciente trabajo en este campo es el desarrollado por Xiong *et al.* (2006). Ellos exploran la utilidad de cuatro técnicas de detección de valores anormales (*outliers*) como métodos de detección de ruido en problemas de clasificación no supervisada, éstas son: el método de detección basado en distancias, la detección basada en agrupamiento (*clustering*) y la detección basada en la medida para cada instancia LOF (*Local Outlier Factor*). La cuarta técnica es propuesta por ellos y se denomina HCleaner (*Hiperclique-based data Cleaner*). Realizan comparaciones experimentales de los cuatro métodos de detección de ruido sobre el impacto en el rendimiento de agrupar los datos usando el algoritmo K-*means*. Una de las limitaciones de este trabajo es que el número de grupos que se usa como parámetro para el algoritmo K-*means*, es el número de clases verdadero que tienen los conjuntos utilizados en sus experimentos. En problemas reales de clasificación no supervisada no es posible saber *a priori* el número de grupos o clases que se van a tener. Finalmente, no se define una supremacía de algunos de los cuatro métodos y se obtienen rendimientos variables de acuerdo al problema analizado.

En la siguiente sección se presenta el algoritmo de detección de ruido en las clases basadas en la calidad de las instancias.

5.3. Método propuesto de detección de ruido en las clases

De acuerdo con la definición de ruido en las clases (Zhu *et al*. 2003), las instancias ruidosas van a ser instancias que se encuentran mal ubicadas dentro de las nubes de puntos que definen cada una de las clases (ver figura 5.3). Esto significa que las instancias ruidosas tendrán valores negativos en las medidas de calidad propuestas y presentadas en el capítulo 3. Sin embargo, algunas instancias ubicadas cerca de la frontera de decisión de dos o más clases, también podrían presentar valores negativos cercanos a cero en cuanto a la evaluación de la calidad de los datos. El algoritmo propuesto identifica las instancias ruidosas y las distingue de las instancias que están en la frontera de decisión. El objetivo es identificar y eliminar las instancias ruidosas, preservando la distribución de las clases y las fronteras de decisión, de forma tal que no se altere la separabilidad de las clases ni el poder discriminante de los algoritmos de clasificación.

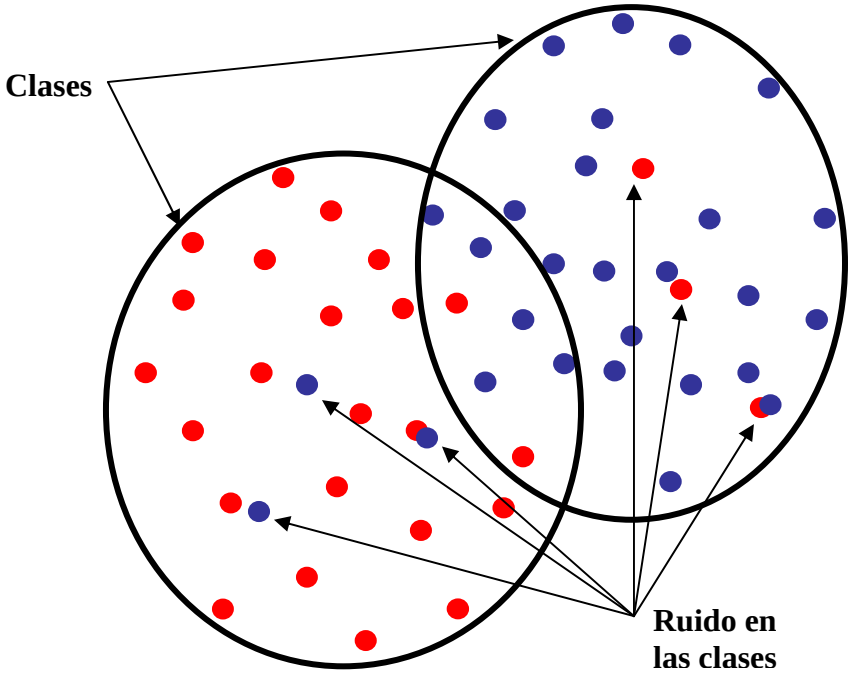

Figura 5.3. Representación gráfica del ruido en las clases

La figura 5.4, muestra el algoritmo **QcleanNOISE** propuesto por nosotros. El algoritmo primero procede a determinar la medida de calidad (Q) para cada instancia del conjunto de entrenamiento. Luego, asigna como candidato a ser ruido a cada instancia cuya medida de calidad sea negativa (Q < 0). El siguiente paso consiste en hallar los vecinos más cercanos de cada instancia candidata a ruido en el conjunto de entrenamiento. Finalmente se realiza un conteo de los vecinos más cercanos que pertenecen a su misma clase. Si la mayoría de sus vecinos más cercanos no son de su misma clase la instancia es considerada como ruido y descartada del conjunto de entrenamiento.

La complejidad en tiempo de computación del algoritmo propuesto es en el peor de los casos de $O(N^2)$. Si observamos el trabajo computacional que realiza el algoritmo, tenemos que tanto el cálculo de la medida de calidad de las instancias y los pasos para determinar cuáles son las instancias candidatas a ruido tienen complejidad lineal $O(N)$. La parte con complejidad cuadrática la determina la búsqueda de los vecinos más cercanos, sin embargo, no se realiza una verificación de todas las instancias en el conjunto de entrenamiento, sino sólo de las que son candidatas a ruido. Por otro lado, en la implementación del algoritmo hemos usado la librería ANN (*Approximate Nearest Neighbor*) (Mount y Arya, 1997), con lo cual, la búsqueda de los vecinos más cercanos se hace en $O(N\log(N))$.

```
QcleanNOISE(Conjunto de Entrenamiento E) {

        For( cada instancia I_i en E)
                Hallar su medida de calidad Q(I_i)
        endFor

        CandNoise = Φ
        For( cada instancia I_i en E)
                if ( Q(I_i) < 0 )
                        CandNoise = CandNoise U { I_i }
        endFor

        For( cada instancia I_i en CandNOISE)
                Hallar sus 5 vecinos mas cercanos (NN) en E
                Count(I) =0
                For( cada NN de I_i )
                        if (Class(NN) ==Class(I_i) ) Count = Count +1
        endFor

        For( cada instancia I_i en CandNOISE)
                if (Count(I) < 3 ) E = E - { I }
        endFor

        Return E
}
```

Figura 5.4. Algoritmo QcleanNOISE

5.4. Evaluaciones experimentales

5.4.1. Metodología

En esta sección presentamos los resultados experimentales para probar el rendimiento del algoritmo propuesto y compararlo con otras alternativas existentes en la literatura. Para verificar el rendimiento en términos de la precisión se usan tres clasificadores: LDA, KNN y RPART. Estos mismos clasificadores se usan para implementar el algoritmo de detección de ruido propuesto por Brodley y Friedl (1996, 1999). Las evaluaciones se realizan usando los conjuntos de datos: *Iris, Breastw, Segment* y *Landsat*.

Para agregar ruido a los conjuntos de datos, adoptamos el mismo esquema utilizado por varios autores (Brodley y Friedl, 1999; Zhu *et al.* 2003; Gamberger *et al*. 1999, 2000). El esquema consiste en: Dado un par de clases (X, Y) y un nivel de ruido w, una instancia etiquetada con la clase X, tendrá una probabilidad del w% de ser cambiada a la clase Y, y viceversa. Según Zhu *et al*. (2003), el uso de este esquema se justifica debido a que en situaciones reales, sólo ciertas clases tienden a presentar problemas de ruido. Queda claro, que de acuerdo a la forma de añadir ruido a las clases, el porcentaje de ruido con respecto al conjunto de entrenamiento completo es menor que el w%. En nuestro experimento, introducimos ruido únicamente a dos clases. En caso de haber más de dos clases se introduce ruido a las clases que tengan una mayor proporción de instancias en el conjunto de entrenamiento.

Para cada experimento usamos validación cruzada con diez particiones y se usa el error promedio como el resultado final. En cada corrida el conjunto de entrenamiento es aleatorizado y siguiendo el esquema de validación cruzada, se divide el conjunto en dos partes. Un 10% de los datos será considerado como conjunto de prueba y el restante (90%), será el conjunto de entrenamiento. Este conjunto de entrenamiento generado en cada partición, es corrompido agregándosele ruido como mencionamos anteriormente y se usa el conjunto de prueba para evaluar el rendimiento del algoritmo propuesto.

De acuerdo a Brodley y Friedl, 1999; Zhu *et al.* (2003), adicionalmente a las medidas del error de clasificación, se usarán las siguientes medidas para evaluar el rendimiento: ER_1, ER_2 y la precisión de la eliminación de ruido (NEP, de sus siglas en inglés). ER_1, ocurre cuando una instancia no ruidosa es considerada ruido; ER_2, ocurre cuando una instancia que es ruido, es considerada una instancia correctamente etiquetada. La precisión mide el porcentaje del número de instancias que han sido catalogadas como ruido en el conjunto de instancias detectadas como ruido. Las fórmulas para cada una de ellas están dadas por las ecuaciones 5.1.

$$NEP = \frac{|D \cap R|}{|D|} \qquad ER_2 = \frac{|\tilde{D} \cap R|}{|R|} \qquad ER_1 = \frac{|D \cap \tilde{R}|}{|\tilde{R}|} \tag{5.1}$$

Donde:
D = Conjunto de instancias que han sido detectadas como ruido y se han eliminado.
R = Conjunto de instancias ruidosas.
$\tilde{D}$ y $\tilde{R}$ representan sus respectivos complementos.

5.4.2. Resultados

La tabla 5.1 muestra los valores para el conjunto de datos Iris, de cada una de las medidas definidas en la ecuación 5.1. para el método de detección de ruido propuesto por nosotros (QcleanNOISE) y las tres alternativas de Brodley and Friedl (1999). Estos valores corresponden al promedio de 15 repeticiones.

El método propuestto QcleanNOISE detecta menos instancias buenas como ruidos (ER_1), que el esquema de eliminación de Brodley que usa la votación por consenso. A bajos niveles de ruido (menos del 10%), los esquemas de eliminación Majority y Consensus, muestran niveles bajos en la identificación de ruido como instancias buenas (ER_2). Pero, a niveles altos de ruido (más de 20%), los niveles del error ER_2 se disparan. El problema de tener niveles altos de ER_1 y ER_2, es que el método está eliminando muchas instancias buenas como si fueran ruido y/o reteniendo muchas instancias ruidosas como si fueran

buenas. Este problema provoca que los algoritmos de clasificación que usen los conjuntos de entrenamientos limpios con altos niveles de ER1 y/o ER_2, tiendan a perder precisión (altos errores de clasificación). Este efecto lo podemos observar en la tabla 5.2. y la figura 5.5. Por ejemplo, el algoritmo de detección propuesto, presenta bajos niveles de ER1 y niveles moderados de ER_2, que se ve reflejado en una mayor precisión para los tres algoritmos de clasificación (ver figuras 5.5, 5.6 y 5.7 y tablas 5.2, 5.3 y 5.4).

En términos de la precisión para detectar el ruido en el conjunto de datos, podemos observar que nuestra propuesta presenta una mayor precisión para todos los niveles de ruido.

Para el caso del conjunto *Breastw*, se observa un comportamiento similar al observado con el conjunto *Iris* (ver tablas 5.5-5.8 y figuras 5.8-5.10), tanto en las medidas de detección de ruido como en el comportamiento de los errores de clasificación. Al igual que en *Iris*, los niveles del ER_1 son bastante bajos (menos del 2%), manteniéndose niveles moderados de ER_2. Nuevamente, los niveles de precisión son bastante altos comparados con las otras alternativas. Con respecto a los errores de clasificación, se observa que consistentemente no se incrementan drásticamente comparados con los métodos de Brodley, que tienden a incrementarse a medida que el nivel de ruido aumenta.

El conjunto de datos *Segment* tiene 7 clases, cada una de las cuales tiene 330 observaciones, bajo el esquema de agregar ruido a las clases, un 5% de ruido agregado a dos clases equivale aproximadamente a 1.5% de ruido sobre el total de instancias en el conjunto de entrenamiento (y un 13 aprox. para el 45% de ruido), esto hace que el impacto del ruido sobre el error de clasificación sea menor, lo que se visualiza en las figuras 5.11-5.13. Se observa una aparente igualdad en el impacto sobre el error de clasificación hasta el 30% de ruido introducido. Sin embargo la tendencia es la misma que se observó en los conjuntos *Iris* y *Breastw*.

En el conjunto de datos *Landsat*, el comportamiento es similar al conjunto de datos *Segment*, ya que *Lansat* tiene 5 clases y las dos clases con mayor número de instancias tienen en conjunto el 47.26% del total de instancias. Por lo tanto, el impacto de introducir ruido, se reduce, aunque la tendencia de los resultados obtenidos es la misma a la observada en los tres conjuntos de datos analizados anteriormente (ver tablas 5.13-5.15 y figuras 5.14-5.16)

Finalmente, de acuerdo a los resultados mostrados, de los tres clasificadores utilizados en estos experimentos, el más sensible a la presencia de ruido en las clases es el clasificador KNN.

Tabla 5.1 Resultados de la eliminación de ruido en el conjunto de datos Iris

Ruido	QcleanNOISE			Single		
	ER1	ER2	NEP	ER1	ER2	NEP
5%	0.14	0.00	97.50	4.44	6.67	55.25
10%	0.22	2.22	97.88	10.12	12.67	49.98
20%	0.29	5.67	98.45	19.46	20.33	50.31
30%	0.37	16.52	98.56	27.75	32.22	50.38
40%	0.44	27.89	98.58	38.22	39.50	50.64
45%	0.59	41.09	98.35	42.68	46.07	49.98

Ruido	Majority			Consensus		
	ER1	ER2	NEP	ER1	ER2	NEP
5%	0.48	0.00	94.60	0.14	1.33	97.50
10%	0.48	0.00	96.67	0.19	5.33	98.10
20%	3.43	2.00	87.94	0.19	25.00	98.91
30%	11.72	10.44	75.94	1.46	41.56	94.55
40%	23.43	21.00	68.65	7.93	55.67	78.37
45%	37.84	35.56	57.31	10.94	66.67	70.22

Tabla 5.2. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Iris usando LDA

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	4.667	2.000	2.622	2.133	2.044
10%	7.022	2.178	2.533	2.267	2.089
20%	10.667	3.689	2.978	4.311	2.133
30%	13.156	8.622	5.289	9.111	2.222
40%	15.467	14.178	11.156	13.511	2.578
45%	22.667	20.400	16.889	19.600	2.311

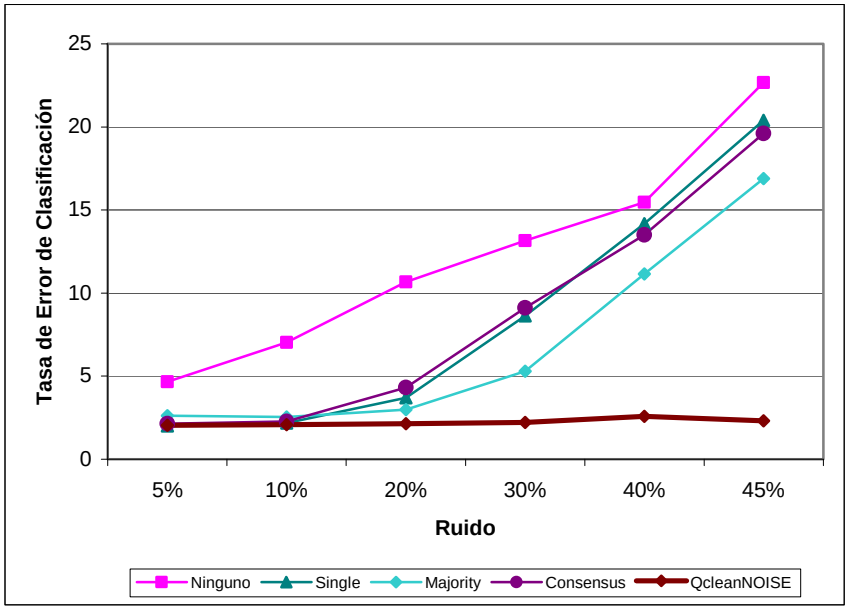

Figura 5.5. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Iris usando LDA

Tabla 5.3. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Iris usando KNN

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	4.133	3.289	3.778	4.133	3.822
10%	5.156	3.378	3.778	4.133	3.644
20%	9.067	5.911	4.133	5.511	3.511
30%	15.244	11.378	7.111	10.667	3.778
40%	22.400	18.000	14.000	16.844	4.444
45%	23.822	21.911	17.733	20.667	4.178

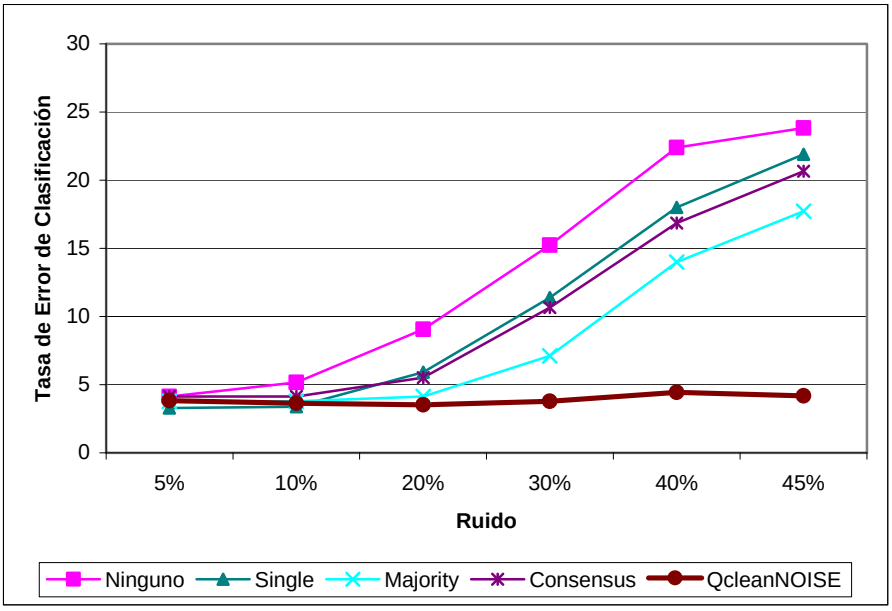

Figura 5.6. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Iris usando KNN

Tabla 5.4. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Iris usando RPART

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	6.578	6.267	5.378	5.644	7.156
10%	7.156	6.400	5.511	5.911	6.400
20%	8.089	6.089	5.689	5.467	5.644
30%	14.444	8.400	6.356	8.489	5.111
40%	22.978	15.600	12.000	16.533	5.467
45%	26.622	21.733	18.711	20.844	5.111

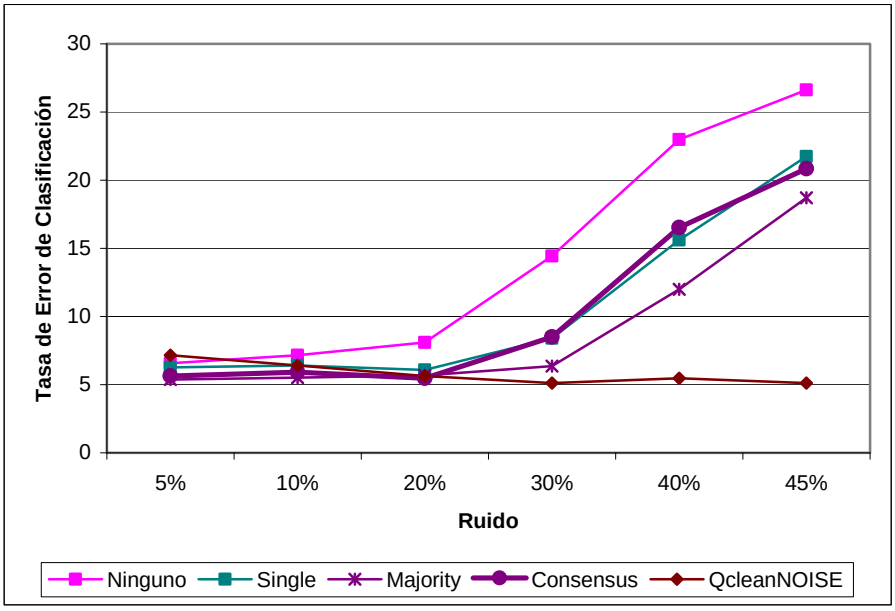

Figura 5.7. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Iris usando RPART

Tabla 5.5. Resultados de la eliminación de ruido en el conjunto de datos Breastw

Ruido	QcleanNOISE			Single		
	ER1	ER2	NEP	ER1	ER2	NEP
5%	1.86	4.64	72.96	6.97	8.24	41.11
10%	1.90	6.76	84.44	9.25	8.92	52.32
20%	1.72	11.73	92.75	14.70	15.10	59.03
30%	1.49	22.49	95.69	23.91	22.71	58.01
40%	1.37	37.01	96.85	34.15	34.51	56.16
45%	1.37	46.50	96.95	41.93	42.15	53.05

Ruido	Majority			Consensus		
	ER1	ER2	NEP	ER1	ER2	NEP
5%	3.63	4.12	58.24	1.62	11.37	74.19
10%	3.90	4.02	73.29	1.55	14.41	86.06
20%	4.69	4.26	83.59	1.46	22.16	92.99
30%	5.92	6.70	87.16	1.66	29.41	94.81
40%	12.00	12.80	82.90	2.44	42.44	93.98
45%	20.98	20.33	75.69	5.46	58.78	85.74

Tabla 5.6. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos *Breastw* usando LDA

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	3.953	4.119	4.256	4.061	4.031
10%	4.139	4.178	4.285	4.119	4.012
20%	4.636	4.392	4.402	4.334	3.982
30%	5.281	5.290	4.549	4.841	3.973
40%	9.898	9.400	8.092	8.658	3.992
45%	18.116	15.832	13.919	14.261	4.021

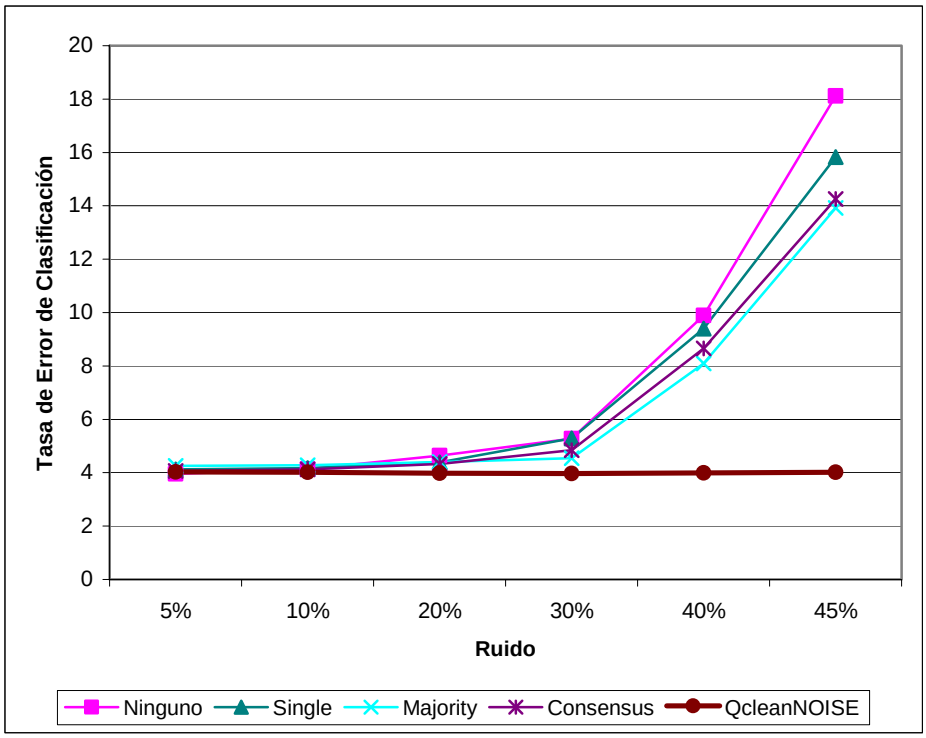

Figura 5.8. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos *Breastw* usando LDA

Tabla 5.7. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos *Breastw* usando KNN

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	3.641	3.328	3.309	3.153	3.367
10%	4.959	3.807	3.504	3.777	3.494
20%	9.419	5.759	3.631	5.281	3.504
30%	16.916	12.143	4.334	9.566	3.690
40%	30.542	27.008	10.259	20.078	3.982
45%	39.639	37.062	21.240	29.771	3.924

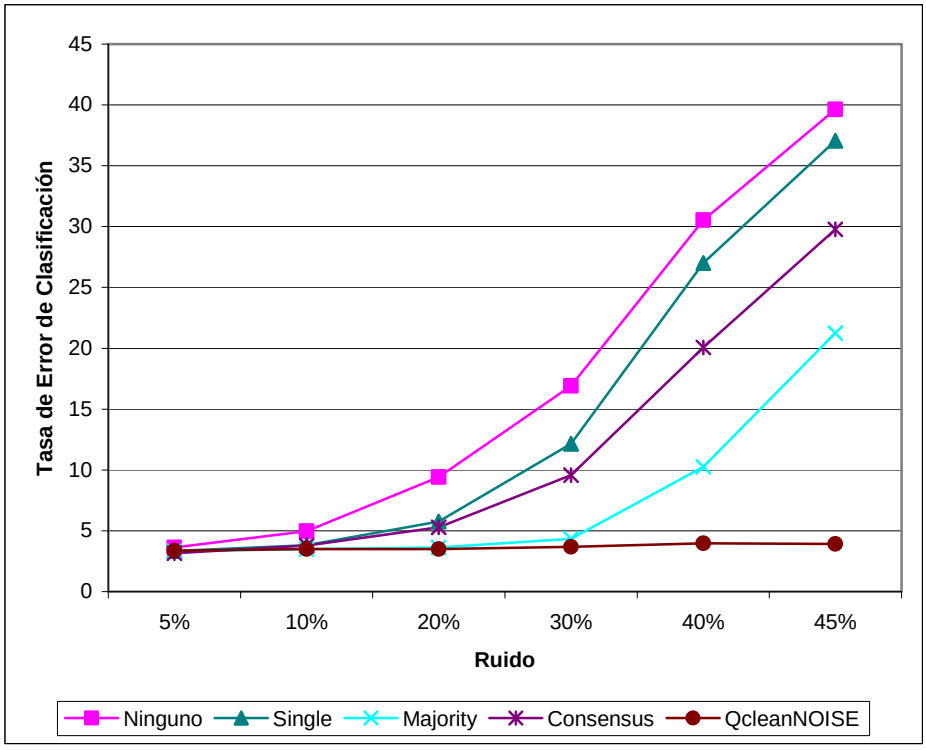

Figura 5.9. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos *Breastw* usando KNN

Tabla 5.8. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos *Breastw* usando RPART

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	5.173	4.822	4.636	5.027	5.085
10%	5.290	5.076	4.744	5.134	5.056
20%	5.671	5.573	4.968	5.437	5.466
30%	7.906	7.350	5.944	6.374	5.486
40%	14.661	19.922	10.298	11.918	5.944
45%	24.090	34.046	20.098	21.386	5.603

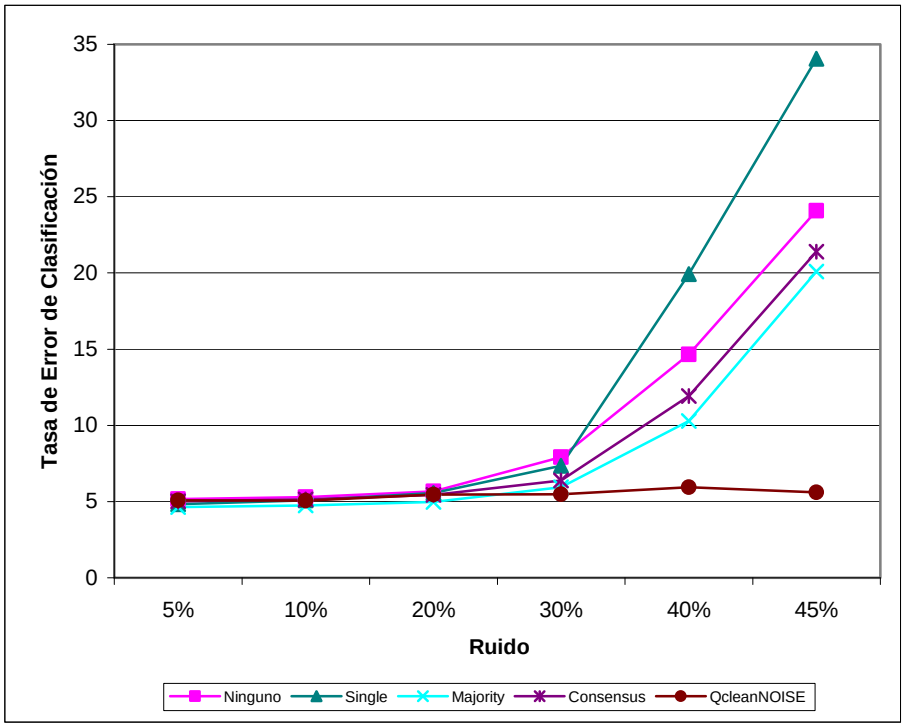

Figura 5.10. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos *Breastw* usando RPART

Tabla 5.9. Resultados de la eliminación de ruido en el conjunto de datos Segment

Ruido	QcleanNOISE			single		
	ER1	ER2	NEP	ER1	ER2	NEP
5%	0.24	0.13	95.23	5.05	2.83	48.49
10%	0.24	0.77	96.31	9.64	8.69	49.34
20%	0.27	6.82	97.26	18.92	20.05	49.57
30%	0.35	19.20	97.32	28.00	30.10	49.75
40%	0.44	33.17	97.09	38.00	38.81	49.79
45%	0.50	42.02	96.94	42.34	44.83	49.69

Ruido	Majority			Consensus		
	ER1	ER2	NEP	ER1	ER2	NEP
5%	0.37	0.00	94.38	0.09	5.66	96.67
10%	0.45	0.10	94.85	0.10	8.18	97.68
20%	0.66	0.15	95.69	0.10	19.50	98.55
30%	3.52	2.86	90.52	0.33	34.44	98.05
40%	15.09	13.86	77.68	2.16	52.15	92.96
45%	29.19	29.52	64.92	6.28	64.53	80.99

Tabla 5.10. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Segment usando LDA

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	8.476	8.909	9.199	8.727	8.918
10%	8.684	8.957	9.286	8.745	8.939
20%	8.861	8.874	9.234	8.710	8.861
30%	9.139	9.087	9.108	8.788	8.805
40%	9.502	9.892	9.433	9.264	9.074
45%	10.403	10.978	10.277	9.978	8.935

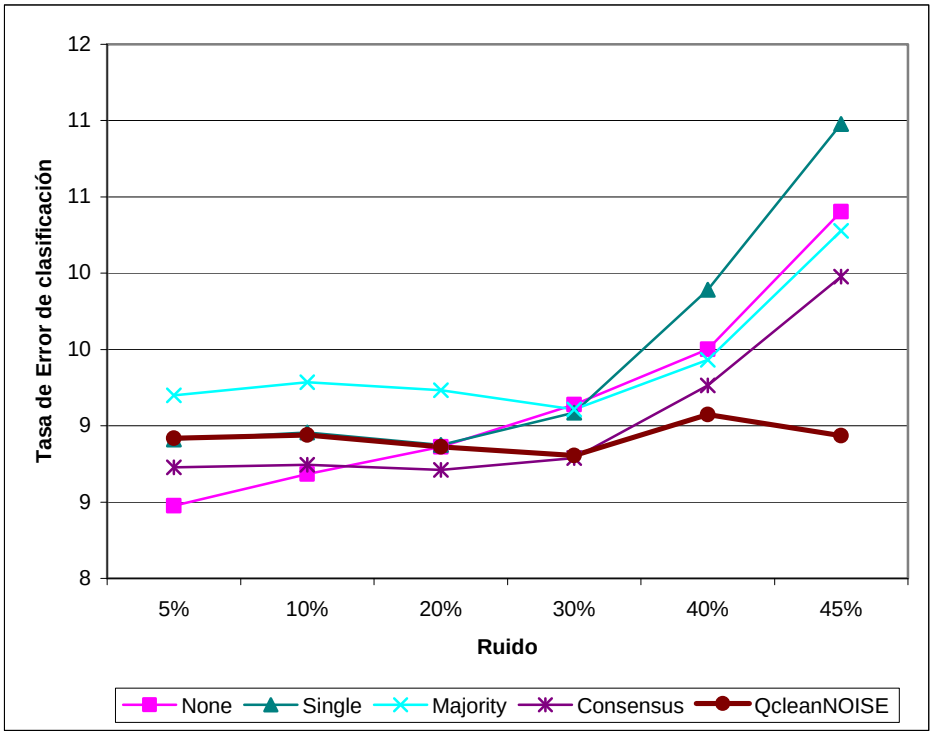

Figura 5.11. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Segment LDA

Tabla 5.11. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Segment usando KNN

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	4.788	5.615	5.039	4.113	5.342
10%	5.442	5.810	4.996	4.247	5.320
20%	6.987	6.468	5.199	4.732	5.446
30%	9.580	9.013	5.299	5.926	5.632
40%	12.481	12.182	6.177	8.069	5.866
45%	14.173	14.238	8.394	10.268	5.861

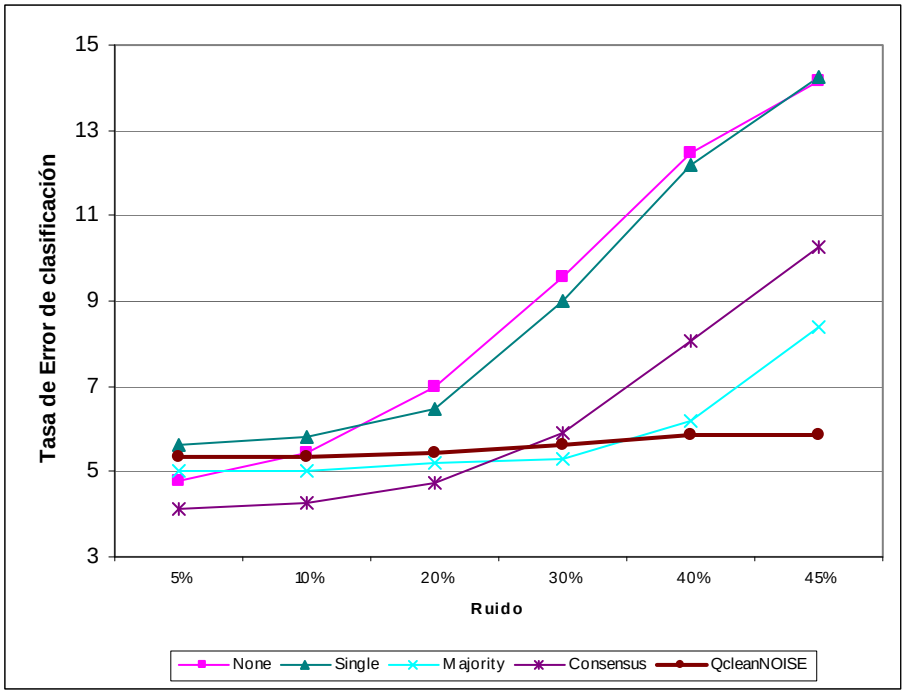

Figura 5.12. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Segment usando KNN

Tabla 5.12. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Segment usando RPART

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	8.390	8.182	8.026	8.139	8.307
10%	8.459	8.338	7.996	8.177	8.571
20%	8.481	8.290	7.827	8.394	8.714
30%	8.792	8.541	8.199	8.481	8.693
40%	9.528	9.935	9.147	8.874	8.749
45%	13.143	12.584	10.290	10.506	8.866

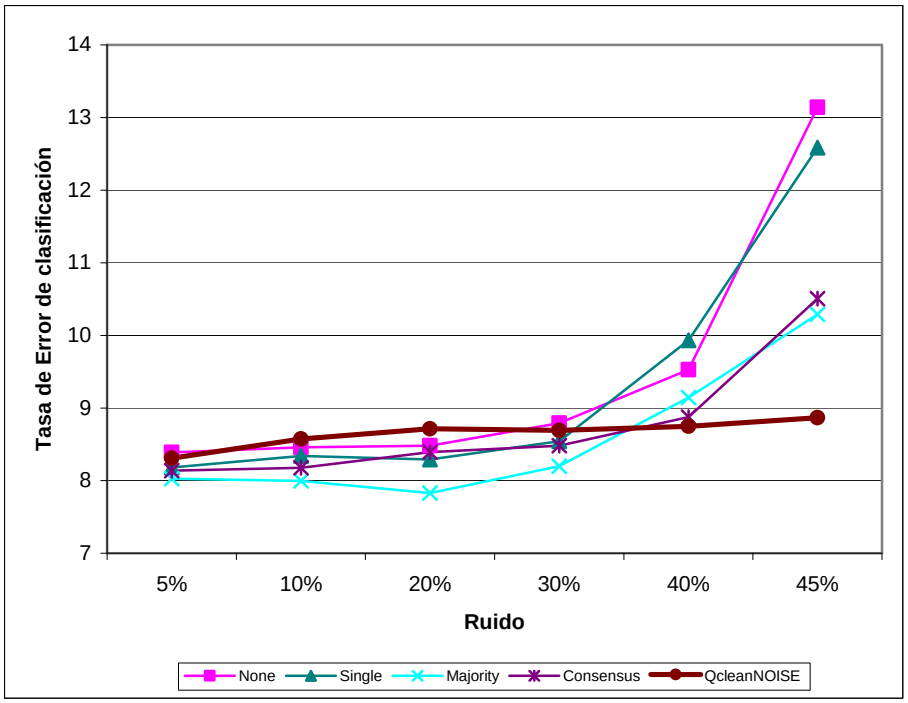

Figura 5.13. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Segment usando RPART

Tabla 5.13. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Landsat usando LDA

		Brodley			
Ruido	Ninguno	Single	Majority	Consensus	QcleanNOISE
5%	16.140	16.252	16.503	16.146	16.053
10%	16.162	16.183	16.357	16.137	16.093
20%	16.370	16.075	16.379	16.003	15.935
30%	16.575	16.326	16.205	16.171	15.925
40%	17.131	17.324	16.398	16.659	15.845
45%	19.239	19.214	17.625	17.930	15.963

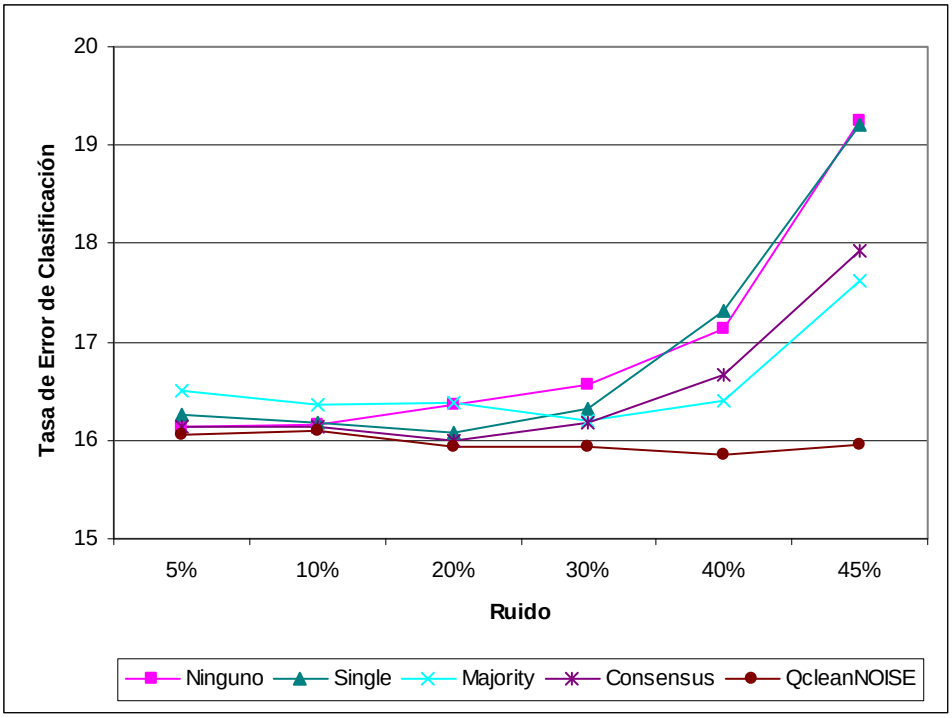

Figura 5.14. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Landsat usando LDA

Tabla 5.14. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Landsat usando KNN

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	9.355	9.756	10.772	9.256	9.346
10%	10.412	9.992	11.018	9.464	9.389
20%	13.918	11.354	10.956	10.835	9.448
30%	18.775	15.385	11.465	13.476	9.697
40%	24.718	22.399	12.389	18.477	9.977
45%	27.922	26.974	18.204	23.630	10.023

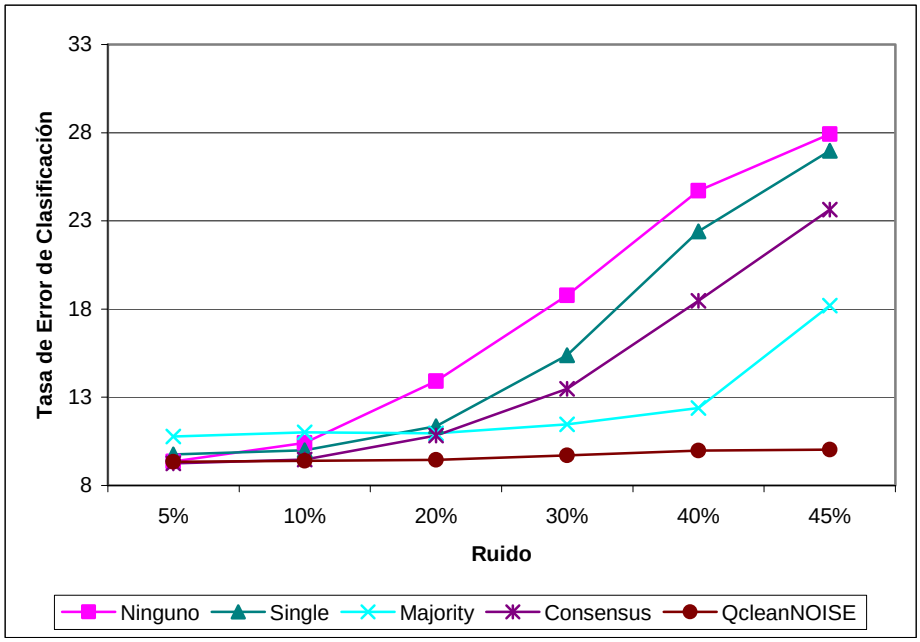

Figura 5.15. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Landsat usando KNN

Tabla 5.15. Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Landsat usando RPART

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	18.337	17.641	18.925	18.347	17.451
10%	18.695	18.163	18.807	18.284	17.395
20%	19.235	18.228	19.192	19.130	17.514
30%	19.453	19.661	19.375	19.360	18.148
40%	21.172	20.531	20.326	20.099	17.756
45%	32.068	24.631	22.859	24.305	18.235

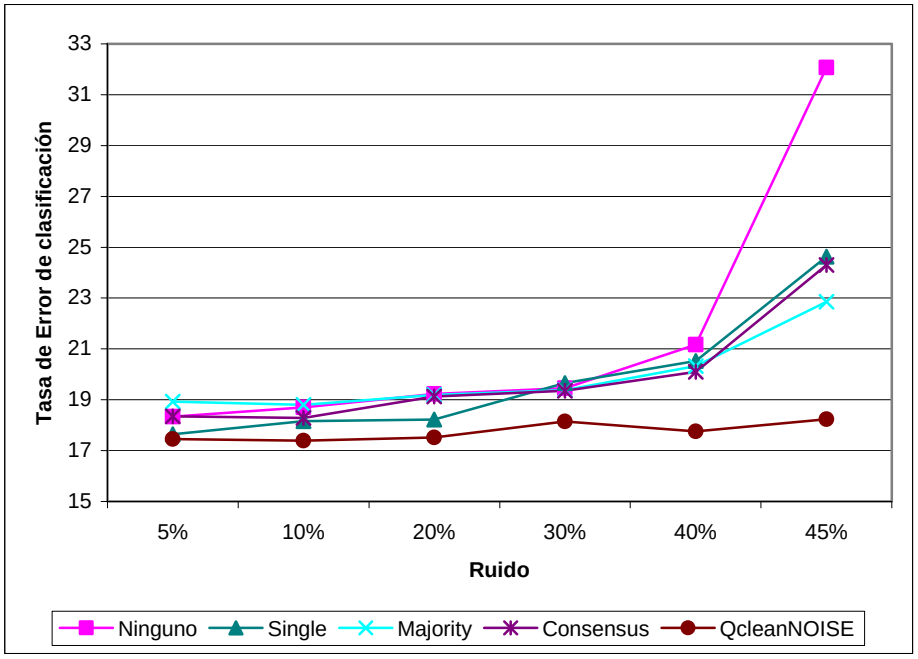

Figura 5.16. Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Landsat usando RPART

Capítulo VI

Selección de instancias

6.1. Introducción

El crecimiento exponencial en la disponibilidad de datos que se generan en las distintas áreas de las ciencias e ingeniería ha creado la necesidad de metodologías que tengan la capacidad para analizar y procesar la información contenida en dichos datos. La selección de instancias emerge como una alternativa para reducir el tamaño del conjunto de datos con la finalidad de hacer posible que los algoritmos de la minería de datos puedan trabajar de manera eficiente y se logre extraer el conocimiento relevante de la inmensa cantidad de datos en la que se encuentra escondida. Sin embargo, el principal problema que enfrentan los algoritmos de selección de instancias es que aún tienen un costo computacional demasiado alto.

Por consiguiente, un importante problema en el proceso KDD es establecer la cota superior para el tamaño del conjunto de datos, necesaria para que los algoritmos de DM, realicen un análisis eficiente y preciso (Vucetic y Obradovic, 2000). Por ejemplo, en muchas aplicaciones incrementar el tamaño del conjunto de datos 10 veces para una posible ganancia de precisión de 1%, no puede justificar los inmensos costos computacionales. Por otro lado, como se ha mencionado anteriormente, grandes conjuntos de datos pueden incrementar la complejidad de los modelos y no permitir su generalización (Oates y Jensen, 1998)

Reduciendo un conjunto grande de datos a un subconjunto representativo y compacto, que permita retener esencialmente el mismo conocimiento extraíble, puede mejorar la velocidad de aprendizaje y reducir los requerimientos de almacenamiento. Adicionalmente, puede permitir la aplicación de algoritmos más poderosos, pero con mayores requerimientos computacionales y que son atractivos para descubrir conocimientos más interesantes en los datos. En este sentido la escalabilidad es un requerimiento clave para KDD y los algoritmos de la minería de datos (Zhu y Wu, 2006; Domingo *et al.*, 2002).

El muestreo progresivo permite mejorar el rendimiento de los algoritmos de selección de instancias cuando el tamaño del conjunto de datos es grande, reduciendo los costos computacionales en espacio y tiempo, ya que no se requiere procesar la totalidad de las instancias en la muestra de entrenamiento para seleccionar las instancias más relevantes. En este trabajo se presentan algunos resultados experimentales y estudios comparativos de los conjuntos de datos disponibles para evaluar la efectividad y eficiencia de la estrategia propuesta.

6.2. Selección de Instancias

Según Liu y Motoda (2002), el resultado ideal de la selección de instancias es un modelo independiente, que permite obtener una muestra mínima de los datos con los que se pueda realizar las tareas de la minería de datos con un mínimo o ningún deterioro en su rendimiento, es decir, para un algoritmo de minería de datos M, su rendimiento P en una muestra s de instancias seleccionadas del total de datos disponibles w, debemos tener:

$$\mathrm{P}(M_s) \cong \mathrm{P}(M_w) \tag{6.1}$$

Por un modelo independiente, debemos entender que para dos algoritmos de DM, M_i y M_j, sea ΔP la diferencia en el rendimiento usando los datos s con respecto a usar los datos w,

$$\Delta P(M_i) \cong \Delta P(M_j) \tag{6.2}$$

El problema de selección de instancias tiene trascendencia debido a las inmensas cantidades de datos y las necesidades crecientes de preparar los datos para las aplicaciones utilizadas para extraer conocimiento. La mayoría de las veces, se tiene que realizar la selección de las instancias para obtener mejoras significantes en los resultados. Liu y Motoda indican que la selección de instancias tiene tres funciones prominentes (Liu y Motoda, 2002):

1. **Habilitar un conjunto de datos**. Como se mencionó antes, el principal problema de muchos algoritmos es que no pueden trabajar con grandes cantidades de datos, o tienen problemas con los tipos y formatos de datos. La selección de instancias hace posible que se puedan ejecutar los algoritmos disponibles de la minería de datos y que puedan trabajar de manera eficiente. Esto es posible debido a que la selección de instancias reduce la cantidad de datos (número de filas).
2. **Enfocar el descubrimiento de conocimiento**. Esta función tiene por objetivo centrar el análisis sobre un aspecto en particular del dominio, esto se debe a que una aplicación es normalmente sólo aproximadamente un aspecto del dominio. Por lo tanto, es natural y sensato enfocarse en la parte pertinente de los datos para la aplicación, de forma tal que la búsqueda se enfoque más y la minería sea más eficaz.
3. **Limpiar el conjunto de datos**. Otro problema que permite solucionar la selección de datos, es la de eliminar los datos irrelevantes, los datos redundantes y reducir el ruido. Al quedarnos con los datos que tienen una mayor calidad obtendremos resultados adecuados y la reducción de los costos computacionales para la minería de los datos.

El proceso de selección de instancias no es gratuito, se tienen que utilizar computadoras, operarios de computadoras y lo más trascendente, invertir tiempo. Por lo tanto, si bien es

cierto que la selección de instancias trae consigo una serie de beneficios, es importante hacer el contraste de lo que se gana contra lo que se pierde debido a la selección de instancias. La evaluación más utilizada es la de comparar los tiempos que involucran el proceso de selección y las tareas de la minería de datos, pero no debe de ser la única, ya que también se tiene que evaluar qué tan eficiente es el proceso de selección de datos y su efecto en los resultados finales. Este problema está firmemente asociado con otros dos problemas: (1) el tamaño de la muestra y (2) la calidad de la muestra seleccionada para preservar el conocimiento de los datos completos (Liu y Motoda, 2002). Lo más importante es la calidad de la muestra seleccionada. En el contexto de la reducción de los datos, se deben reducir los mismos pero manteniendo la calidad de los datos completos. De esta forma, se tiene que enfrentar el problema de hallar el equilibrio entre el tamaño de la muestra y su calidad al seleccionar las instancias (muestra representativa). En este sentido, la selección de instancias es un problema de optimización que intenta mantener la calidad de los datos completos en la muestra seleccionada mientras se minimiza el tamaño de la muestra (Liu y Motoda, 2002).

6.3. Clasificación de los métodos de selección de instancias

De acuerdo a Wilson y Martinez (2000), el proceso de edición puede llevarse a cabo siguiendo tres procesos de búsqueda de las mejores instancias: Búsqueda hacia adelante, búsqueda hacia atrás y por lotes.

Búsqueda hacia adelante: En este tipo de búsqueda (ver figura 6.1) se parte de un conjunto vacío S y en cada paso se añade a S la instancia que satisface cierto criterio de selección de instancias. Este proceso de búsqueda es sensible al orden en que son evaluadas las instancias en el conjunto de entrenamiento, ya que las primeras instancias en aparecer, tendrán una probabilidad mayor de pertenecer a S, que las últimas instancias. Esto se debe a que las últimas instancias en probarse, puede que ya estén representadas por alguna instancia que ingresó anteriormente. En este sentido, puede verse dañada la precisión del clasificador si las últimas

instancias son más representativas que las primeras. La principal ventaja de este tipo de búsqueda es que resulta ser más rápida ya que consume menos recursos de almacenamiento durante el proceso de entrenamiento del clasificador en comparación a las estrategias no incrementales.

Entre las principales desventajas de la búsqueda incremental, como lo mencionamos anteriormente, está la sensibilidad al orden en que se presentan las instancias. Por otro lado, las primeras decisiones se basan en muy poca información y por tanto estas decisiones son propensas a errar en la clasificación. Por esta razón, algunos métodos incrementales realizan una fase denominada de grupo inicial, que consiste en partir de un determinado número de objetos en el conjunto S y después aplicar la búsqueda incremental (Wilson y Martinez, 2000).

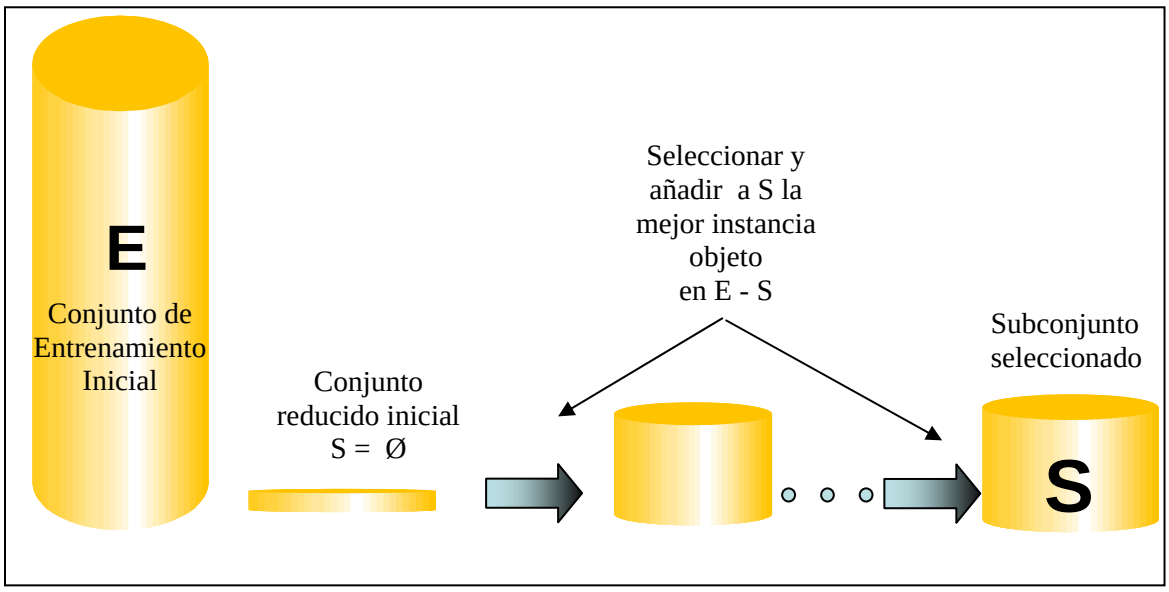

Figura 6.1. Búsqueda hacia adelante

Búsqueda hacia atrás: Esta estrategia comienza igualando el conjunto de entrenamiento al conjunto de instancias seleccionadas $S = E$ y en cada paso se

determina la instancia a eliminar de S de acuerdo al criterio de selección de instancias (ver figura 6.2). Al igual que en la búsqueda incremental, es importante el orden en que las instancias son evaluadas, pero a diferencia de las técnicas incrementales todos los objetos parcialmente almacenados están disponibles en todo momento para examinar cual de ellos resulta conveniente eliminar.

Según Wilson y Martinez (2000), la principal ventaja en este tipo de estrategia es que se obtiene una mayor reducción del E y normalmente con el subconjunto S que se obtiene, se logra una mayor precisión de clasificación con respecto a la obtenida con la muestra original.

La desventaja que presenta esta búsqueda, es que los costos computacionales son más altos que el enfoque de búsqueda hacia adelante, ya que, por ejemplo, para encontrar similitud entre una instancia y el subconjunto S, la estrategia hacia atrás lleva a cabo N comparaciones, donde N, es el número de instancias en S. Por otro lado, la estrategia incremental realiza menos cálculos (cero inicialmente y posteriormente solo una fracción de N).

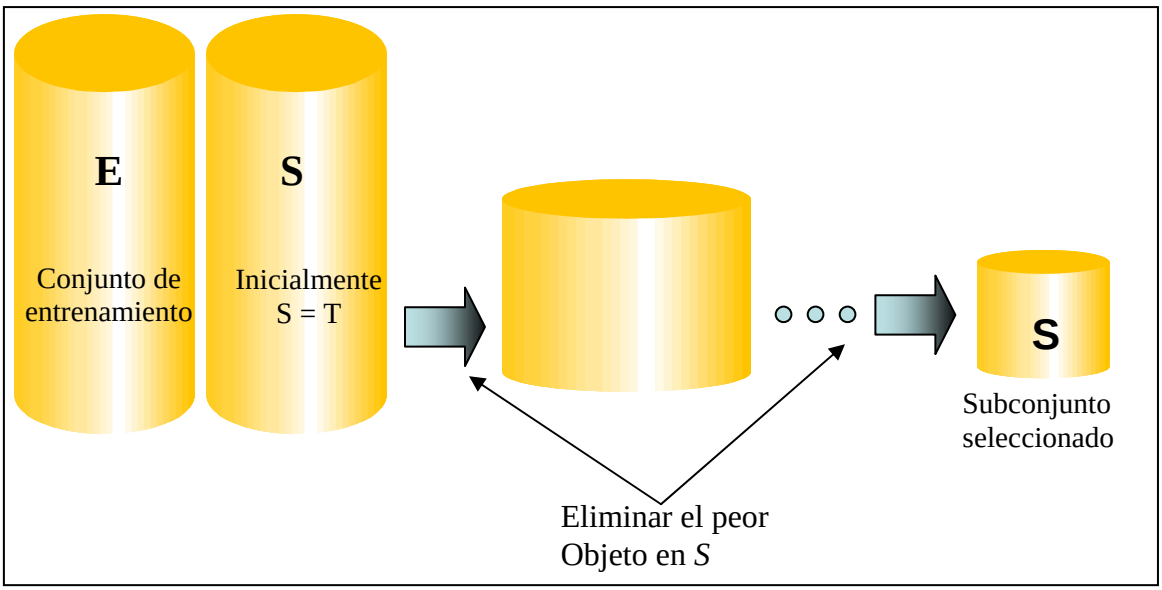

Figura 6.2. Búsqueda hacia atrás

Estrategia por lotes: Otra forma de llevar a cabo la selección de instancias, consiste en identificar y marcar las instancias que no satisfacen el criterio de selección, las cuales no serán consideradas en el subconjunto S. Finalmente se eliminan tales instancias, es decir, no se elimina solo un objeto, sino grupos de estos. Al igual que la búsqueda hacia atrás, esta técnica tiene altos costos computacionales.

Según Brighton y Mellish (2002), los métodos de selección de instancias también pueden clasificarse de acuerdo al efecto que causa la eliminación de las instancias. De esta forma podemos distinguir que los métodos de selección de instancias se clasifican en:

- **Incremento de la competencia**: Este método se enfoca a eliminar aquellos objetos, los cuales, al ser descartados, la precisión en los resultados de clasificación se incrementa. Normalmente, esta técnica elimina objetos considerados como ruido.

- **Preservación de la competencia**: Esta técnica elimina objetos superfluos, es decir, aquellos objetos cuya eliminación no provoca un decremento en la precisión de los resultados de clasificación.

- **Esquema híbrido**: Se deriva de los dos esquemas anteriores y se encarga de abordar ambos problemas a la vez.

6.4. Algoritmos de selección de instancias

En esta sección presentamos brevemente la descripción de los más conocidos algoritmos de selección de instancias.

Algoritmo CNN (*Condensed Nearest Neighbour*): Introducido en 1968 por Hart. Este algoritmo busca un subconjunto S del subconjunto del entrenamiento E, de forma que cada instancia en S está más cercana a una instancia de su clase, que a un miembro de otra clase. Este método comienza seleccionando de manera aleatoria un objeto correspondiente a cada una de las distintas clases y estos objetos se añaden a S, el cual inicialmente es un conjunto vacío. Posteriormente, cada objeto en E es clasificado empleando únicamente los objetos de S; cuando un objeto es clasificado erróneamente entonces este se añade a S, para garantizar que será clasificado correctamente. El proceso se repite hasta que no existan objetos en E que sean clasificados de manera errónea.

Esta técnica es sensible al ruido, debido a que las instancias ruidosas se clasifican erróneamente por sus vecinos y de esta forma, las instancias ruidosas se agregan a S, lo cual provoca dos inconvenientes: el primero es que no se logra una reducción considerable de la muestra, ya que los objetos ruidosos son innecesarios pero aún siguen presentes en el conjunto de entrenamiento reducido y el segundo inconveniente es el efecto negativo que el subconjunto resultante causa en los resultados de clasificación, debido a que los objetos ruidosos no aportan información relevante al clasificador y pueden provocar una reducción en la precisión. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^3\right)$, donde N es el tamaño del conjunto de entrenamiento.

Algoritmo SNN (*Selective Nearest Neighbor*): Ritter *et al*. (1975) proponen la regla selectiva del vecino más cercano (SNN), la cual es una extensión del algoritmo CNN. De acuerdo a SNN, los objetos de E estarán más cerca a objetos de S con la misma clase que a los objetos de E con distinta clase, además, SNN garantiza encontrar un conjunto mínimo que satisface tal condición. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^3\right)$.

Algoritmo RNN (*Reduced Nearest Neighbor*): Introducido por Gates (1972). Realiza una extensión de búsqueda hacia atrás de CNN con la regla del vecino más cercano reducido. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^3\right)$.

Algoritmo ENN (*Edited Nearest Neighbor*): Presentado por Wilson (1972). La idea principal de ENN consiste en remover una instancia si los k vecinos más cercanos no lo clasifican correctamente. Esta técnica suele emplearse para filtrar el ruido en un conjunto de datos, ya que, se eliminan a aquellas instancias (ruidosas) en vecindades de instancias que corresponden a la misma clase. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^2\right)$.

Algoritmo RENN (*Repeated ENN*): También propuesto por Wilson (1972), es una variante de ENN, que consiste en aplicar el algoritmo ENN de manera repetida hasta que todas las instancias en S tengan la misma clase que la mayoritaria de sus k vecinos más cercanos. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^2\right)$.

Algoritmo Multiedit: Propuesto en 1982 por Devijver y Kittler, consiste en una modificación del algoritmo ENN que garantiza la independencia estadística entre las instancias retenidas en el conjunto de datos S. Se realiza una partición del conjunto inicial para obtener m muestras. Posteriormente se procede a aplicar el algoritmo ENN sobre cada una de las particiones obtenidas. Finalmente, el conjunto S estará formado por las instancias retenidas en las m particiones. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^2\right)$.

Algoritmo All k-NN: Es una extensión de ENN realizada por Tomek (1976) con el método de edición por lotes. Consiste en que para cada instancia en S, se determina como lo clasifican sus i vecinos más cercanos ($i = 1, 2, \cdots, k$, con k = número de vecinos), si estos lo clasifican erróneamente, entonces a la instancia se le asigna una marca, una vez

que se han analizado todas las instancias de la muestra, se eliminan las instancias que resultaron marcadas. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^2\right)$.

Algoritmo IB2 (Kibbler y Aha, 1987): Este algoritmo inicia con el conjunto S vacío, luego, las instancias en E que son clasificadas erróneamente, por las instancias en S se van añadiendo a S. El problema de este tipo de estrategia es que resulta ser sensible al ruido, pues en base a la regla que sigue, almacena instancias ruidosas, ya que, por su naturaleza, este tipo de instancias suelen clasificarse de manera incorrecta. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^2 \log N\right)$.

Algoritmo IB3: Elaborada en 1991 por Aha *et al.*, es una extensión de IB2, en el cual nuevamente se inicia con el conjunto S vacío, para luego, analizar los resultados de clasificación antes de eliminar una instancia ruidosa. De esta forma, el algoritmo agrega instancias de E a S, asegurándose de que sólo se añade instancias mal clasificadas consideradas como aceptables. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^2 \log N\right)$.

Wilson y Martínez (2000) propusieron los métodos de selección de instancias hacia atrás DROP (*Decremental Reduction Optimization Procedure*). Estos métodos basan su regla de selección en términos del concepto de asociada. La asociada de una instancia I es aquella instancia que tiene a I como uno de sus k vecinos más cercanos. La complejidad asintótica en tiempo de los algoritmos DROP1-5 es $O\left(N^3\right)$.

Algoritmo DROP1: Se elimina la instancia I de S si sus instancias asociadas en S se clasifican correctamente sin I, es decir, bajo este criterio, la ausencia de I no afecta en la clasificación.

Algoritmo DROP2: En este algoritmo se verifica el efecto que causa la eliminación de la instancia I en las instancias del conjunto de entrenamiento E, es decir, DROP2 elimina la instancia I de S si las asociadas que I tiene en E clasifican correctamente sin I.

Algoritmos DROP3 y DROP4: Aplican un filtrado de ruido (similar al método ENN) antes de comenzar el proceso de selección de instancias. La diferencia entre ambos es el criterio empleado en la etapa de filtrado, ya que DROP4 antes de eliminar la instancia ruidosa, verifica el impacto de clasificación provocado al no considerar tal instancia para determinar si será o no eliminado. Finalmente, el método **DROP5** modifica DROP2 de tal manera que comienza por eliminar instancias que se encuentran cerca de los enemigos más cercanos (instancias cercanas con distinta clase).

Algoritmo ICF *(Iterative Case Filtering)*: Desarrollado por Brighton y Mellish (2002), este algoritmo hace uso de los conceptos de alcance y cobertura de la instancia I. La regla de selección es la siguiente: eliminar aquellas instancias tales que el tamaño del alcance es mayor que el de cobertura, lo cual quiere decir que una instancia I será eliminada cuando mediante otras instancias se puede obtener la información que pudiera proporcionar. Como etapa inicial, ICF filtra la muestra empleando ENN. La complejidad asintótica en tiempo de este algoritmo es $O\left(N^2\right)$.

Una excelente fuente de resultados en términos del porcentaje de reducción y precisión que se obtiene con la mayoría de los algoritmos descritos anteriormente aplicados a 31 conjuntos de datos disponibles en el repositorio de la UCI se encuentra en el trabajo de Wilson y Martinez (2000).

6.5. Muestreo progresivo

El muestreo progresivo, es un muestreo secuencial en el que inicialmente se selecciona una muestra pequeña n_0 y luego se incrementa progresivamente su tamaño hasta obtener un tamaño n_{opt}. El componente central es una secuencia de muestras:

$$S = \{n_0, n_1, n_2, \cdots, n_k\}, \text{ donde } n_i < n_j \text{ para i<j.} \qquad (6.3)$$

Cada n_i representa un tamaño de muestra que se puede probar mediante un algoritmo inductivo. En este caso el criterio PCE utilizado es el porcentaje de aciertos. Existen dos variantes del muestreo progresivo: el muestreo progresivo aritmético (John y Langley, 1996), y el muestreo progresivo geométrico (Provost, *et al*., 1999).

Muestreo progresivo aritmético

En muestreo progresivo aritmético (John y Langley, 1996), la secuencia de muestras está definida por:

$$S = n_0 + i * n_0 = \{n_0, n_0 + n_0, n_0 + 2n_0, \cdots, N\} \qquad (6.4)$$

John y Langley (1996), realizan una comparación entre el muestreo por lotes, que denominan muestreo estático, con su propuesta, que consiste en ir incrementando el tamaño de la muestra a una razón aritmética, hasta obtener el tamaño de muestra óptimo, bajo el criterio PCE.

En sus experimentos ellos utilizaron 11 conjuntos de datos depositados en la UCI. Debido a que muchos de los conjuntos de datos usados fueron muy pequeños, ellos primero replicaron cada caso 100 veces para simular un conjunto de datos grande. El conjunto de datos aumentado se utilizó para generar un conjunto de muestras cuyo tamaño fue sistemáticamente incrementado en 100 instancias entre muestras, de cada iteración (p.e. 100, 200, 300,…, n_{opt}.). Utilizaron el clasificador bayesiano para generar un modelo para cada muestra. Los modelos basados en estas muestras fueron aplicados a unos conjuntos de pruebas previamente seleccionadas, para evaluar su precisión. Las limitaciones de este estudio incluyen el hecho de que los resultados fueron generados por replicación de datos de conjunto de datos pequeños y que los modelos que fueron comparados utilizando sólo el clasificador bayesiano.

Muestreo progresivo geométrico

Provost *et al.* (1999) proponen otro esquema del muestreo progresivo en el que el incremento del tamaño de la muestra se realiza a una razón geométrica. Con esta propuesta ellos buscan superar la limitación en el número excesivo de iteraciones que tiene que realizarse cuando se utiliza del muestreo progresivo aritmético. La secuencia de muestras, se define como:

$$S = a^i * n_0 = \left\{ a * n_0,\ a^2 * n_0,\ a^3 * n_0, \cdots, N \right\}, \qquad (6.5)$$

donde a es una constante diferente de 1 y determina la rapidez con que incrementamos el tamaño de muestra, el cual se va probando mediante un algoritmo inductivo en cada iteración para determinar el tamaño óptimo.

El muestreo progresivo, se diseñó con la finalidad de aumentar la velocidad de aprendizaje inductivo proporcionando aproximadamente la misma exactitud que se obtiene con los datos completos, usando una muestra significativamente más pequeña.

La figura 6.3 muestra el comportamiento típico de la precisión de un algoritmo de minería de datos con el incremento en el tamaño del conjunto de entrenamiento en términos de la precisión. En general, estas curvas tienen un crecimiento acelerado al inicio, seguida de una zona de menor crecimiento para terminar en una zona en la que la curva es aplanada (casi horizontal). El tamaño del conjunto de entrenamiento en donde se inicia la zona aplanada (marcada con una línea vertical roja en la figura 6.3), es a la que nos referimos como tamaño mínimo ($N_{\min}$), que representa el tamaño más pequeño del conjunto de entrenamiento, después del cual no es posible mejorar la precisión del algoritmo agregando datos adicionales. Los modelos construidos con tamaños de muestra menores a $N_{\min}$ tienen menos

precisión, mientras que los modelos construidos con tamaños más grandes tendrán la misma precisión.

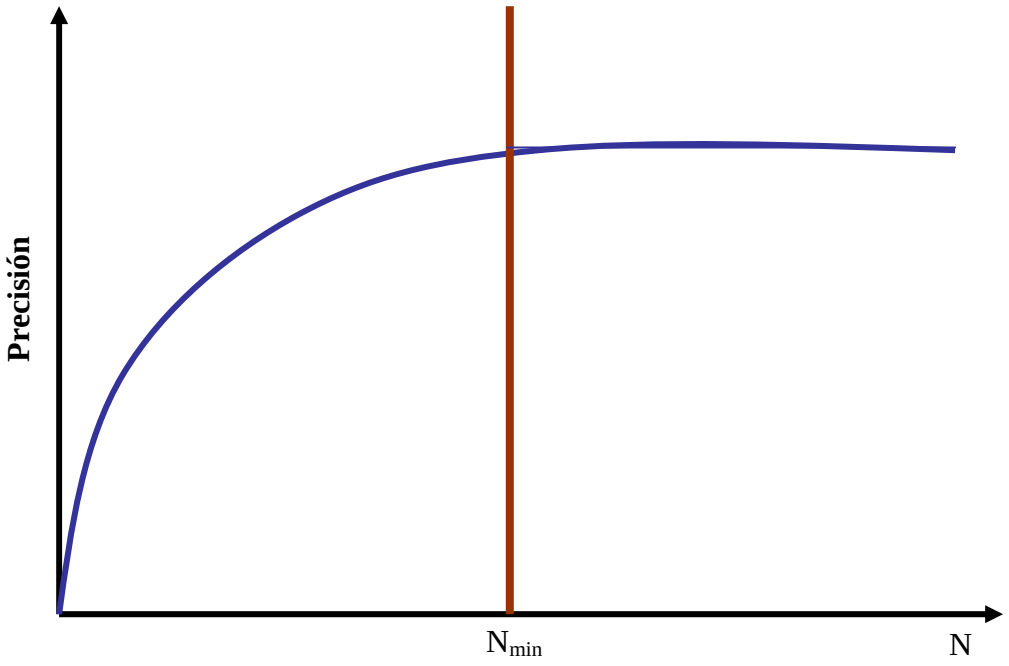

Figura 6.3. Curva de aprendizaje y muestreo progresivo

La idea del muestreo progresivo es detectar la convergencia en la curva de aprendizaje (N_{min}), para seleccionar de manera eficaz el subconjunto de entrenamiento con el cual se van a construir modelos con mayor precisión. El muestreo progresivo requiere de la definición de por lo menos tres componentes principales:

i. La secuencia de muestreo
ii. El conjunto de datos inicial
iii. Criterio de parada

La muestra inicial, puede constituir un elemento importante a considerarse, y que no se tiene en consideración en la propuesta original. Gu *et al*., (2001), tratan este problema y determinan un tamaño óptimo de la muestra inicial utilizando una medida de ajuste de la muestra al conjunto completo, ellos hacen uso de la medida de información de Kullback's (Duda *et al*., 2000), esta propuesta es interesante en el sentido que para determinar la muestra inicial no hace uso del clasificador para medir la calidad de la muestra seleccionada. Sin embargo, los costos computaciones son muy altos y poco alentadores, ya que la ganancia en términos de la convergencia no justificaría, incrementar el uso de los recursos informáticos.

Vucetic y Obradovic (2000), proponen el uso del muestreo progresivo para la reducción del tamaño de conjunto de datos para KDD en grandes bases de datos distribuidas. La propuesta se basa en la reducción de los datos en cada sitio y la construcción de modelos locales, los cuales posteriormente van a ser combinados en un meta-modelo (Stolfo *et al*. 1997). La ventaja de usar meta-modelos es que la construcción de modelos locales y su integración es computacionalmente más eficiente que mover grandes cantidades de datos en una memoria centralizada para construir modelos de aprendizaje global. El objetivo de su propuesta, es reducir el conjunto de datos de tamaño N a N_{min}, de forma que al usar el mismo algoritmo de minería de datos, se pueda extraer el mismo conocimiento, con una pérdida de a lo más α%.

6.6. Algoritmo ProgCNN (Progresivo CNN)

En esta sección proponemos el algoritmo ProgCNN, el cual combina el muestreo progresivo y el algoritmo de selección de instancias CNN, descrita en la sección 6.3. Como se ha mencionado anteriormente, el problema de muchos métodos de selección de instancias tiene que ver con la complejidad computacional en espacio y tiempo. A medida que en el conjunto de entrenamiento se incrementa el número

de instancias, los algoritmos tienden a perder eficiencia y muchas veces no logran ejecutar su trabajo.

Con el algoritmo ProgCNN (ver figura 6.4), se reduce el espacio de búsqueda ya que no se evalúan todas las instancias del conjunto de entrenamiento, para obtener el subconjunto de instancias con una precisión similar al que se obtiene utilizando el conjunto completo. Al usar esta estrategia, se obtienen dos beneficios relevantes: Un mayor porcentaje de reducción respecto al método clásico y un ahorro significativo en cuanto al costo computacional.

Los parámetros de este algoritmo son: el conjunto de entrenamiento (E), la base de la progresión geométrica (α) y la proporción de las instancias en el conjunto de entrenamiento que se van a utilizar como muestra inicial (β).

ProgCNN(Conjunto de Entrenamiento E, a, β) {

- sea i=0
- Sea S = Φ (subconjunto inicial vacío)
- $n_0 = \beta N$ (tamaño de muestra inicial)

1. Seleccionar una muestra inicial de tamaño n_0
2. A la muestra de tamaño n_0 aplicar el algoritmo CNN
3. Las instancias elegidas en el paso 2 se agregan a S
4. Hallar Acc[0], teniendo a S como conjunto de entrenamiento.
5. While($n_0 a^{i+1} < N$) do
6. i= i + 1
7. Seleccionar una muestra de tamaño $n_i = n_0(a^i - a^{i-1})$
8. A la muestra de tamaño n_i aplicar el algoritmo CNN
9. Las instancias elegidas en el paso 8 se agregan a S
10. Usando un algoritmo de clasificación, hallar Acc[i], teniendo a S como conjunto de entrenamiento.
11. Calcular ls1 <- acc[i-1]+sqrt(acc[i-1]*(1-acc[i-1])/n)
12. if((Acc[i] < ls1) & (Acc[i] > = Acc[i-1])) Return S
    else ir al paso 1.

Return S

Figura 6.4. Algoritmo ProgCNN

6.7. Evaluaciones experimentales

6.7.1. Metodología

Se prueba el rendimiento del algoritmo **ProgCNN**, comparado con el algoritmo CNN clásico. Para verificar el rendimiento en términos de la precisión se usan tres clasificadores: LDA, KNN y RPART. Las evaluaciones y comparaciones se realizan usando los conjuntos de datos: *Bupa*, *Segment*, *Abalone*, *Landsat*, *Waveform*, *Penbased y Shuttle* (ver anexo A).

Para cada experimento usamos validación cruzada con diez particiones y se usa el error promedio como el resultado final. En cada corrida el conjunto de entrenamiento es aleatorizado y siguiendo el esquema de validación cruzada, se divide el conjunto en dos partes. Un 10% de los datos será considerado como conjunto de prueba y el restante (90%), será el conjunto de entrenamiento. La selección de instancias se realiza sobre este conjunto de entrenamiento generado en cada partición. Esto es, se aplica cada unos de los algoritmos CNN (ó ProgCNN) para obtener el subconjunto de instancias con los cuales se realiza la clasificación del conjunto de prueba. Este mismo esquema se repite hasta recorrer la totalidad del conjunto de entrenamiento y se repite 10 veces.

Los parámetros utilizados para el algoritmo ProgCNN son: $\alpha = 2$ y $\beta = 0.05$. Para evaluar y realizar las comparaciones, se calcula el error de clasificación, la tasa porcentual de reducción y finalmente se reporta el costo computacional medido en segundos.

6.7.2. Resultados experimentales

Las tablas 6.1 y 6.2, muestran los resultados de la selección de instancias usando los dos algoritmos **CNN** clásico y **ProgCNN**, para el algoritmo LDA. Podemos ver en la tabla 6.1, que en promedio la tasa de reducción aumenta de un 72% con el método clásico a un 78% con el método progresivo propuesto. En términos de los errores de clasificación, se observa

que en promedio son comparables con una pérdida de precisión con respecto al uso de los datos completos de alrededor 1.5% aproximadamente.

Las tasas de reducción tienen un comportamiento similar al del clasificador LDA, en el caso de los clasificadores KNN y RPART (ver figuras 6.3 y 6.5). Con respecto a los niveles de precisión, en el caso del clasificador KNN, los resultados son similares para el algoritmo CNN y ProgCNN. En este caso, la pérdida de precisión con respecto a los datos completos que se observa es de aproximadamente 3%. Para el clasificador RPART, se observa una diferencia entre los errores promedios obtenidos por CNN y ProgCNN, pero es producto de que para el conjunto *segment*, se observó un incremento inusual en el error de clasificación al reducir el conjunto de entrenamiento usando CNN.

Con respecto a estos resultados, se confirma que el algoritmo CNN es muy sensible al ruido. Por ejemplo, si observamos las tablas 6.1, 6.3, y 6.5, podemos notar que para el caso de los conjuntos *bupa* y *abalone* las tasas de reducción son las menores y se observa también que las tasas de error de clasificación sufren un aumento notable con respecto a los errores de clasificación usando el conjunto de entrenamiento completo.

Las tablas 6.2, 6.4 y 6.6, nos muestran los costos computacionales de cada algoritmo de selección de instancias, para los clasificadores LDA, KNN y RPART, respectivamente. Existe una notable disminución en cuanto al tiempo de procesamiento cuando se usa el método progresivo de CNN. En promedio para los conjuntos mostrados en este estudio, el tiempo en segundos (CCS) para el algoritmo CNN usando LDA es de 67183 y disminuye a 4140 segundos con ProgCNN. Para el algoritmo CNN usando KNN es de 78188 y disminuye a 4140 segundos al usar ProgCNN. Finalmente, para el algoritmo CNN usando RPART es de 70233 y disminuye a 4183 segundos cuando se usa ProgCNN.

Por otro lado, si vemos el comportamiento conjunto a conjunto, debemos notar que los conjuntos de datos más ruidosos van a tender a tardarse más tiempo en procesarse. Tal es el caso, por ejemplo, del conjunto de datos Waveform, el cual, a pesar de no ser el

conjunto más grande, tiene un mayor tiempo de procesamiento, para seleccionar las instancias.

Cano *et al.* (2005) utilizarón la estrategia de estratificación para mejorar la escalabilidad de los algoritmos de selección de instancias. En su propuesta, procedierón a realizar una partición del conjunto de entrenamiento (estratificación), utilizando diferentes números de subconjuntos (entre 10 y 30 estratos para conjuntos grandes), luego realiza la selección de instancias en cada estrato usando un algoritmo de selección de instancias y finalmente se unen las instancias elegidas de cada estrato y de esta forma obtener las instancias seleccionadas del conjunto de entrenamiento. Sus resultados experimentales muestran una ganancia en términos de tiempo de procesamiento, sin embargo, se observa que disminuye el porcentaje de reducción de los conjuntos de datos y también se produce una pérdida en el nivel de precisión.

Tabla 6.1. Error de clasificación y tasa de reducción usando el clasificador LDA

Conjunto	Sin sel. Inst.		CNN		ProgCNN	
	Error	Reduc.	Error	Reduc.	Error	Reduc.
BUPA	32.03%	0%	33.62%	40	35.51%	63
SEGMENT	8.53%	0%	7.32%	87	8.03%	83
ABALONE	32.03%	0%	35.96%	41	35.98%	61
LANDSAT	16.07%	0%	15.69%	81	15.48%	82
WAVEFORM	13.93%	0%	15.72%	57	15.68%	66
PENBASED	12.42%	0%	14.86%	96	14.67%	94
SHUTTLE	5.61%	0%	9.36%	96	7.51%	99
Promedio	**17.23%**	**0.00%**	**18.93%**	**71**	**18.98%**	**78**

Tabla 6.2. Tasa de reducción y costo computacional usando el clasificador LDA

Conjunto	CNN		ProgCNN	
	Reducción	CCS	Reducción	CCS
BUPA	40	61	63	30
SEGMENT	87	1133	83	315
ABALONE	41	30125	61	2000
LANDSAT	81	91761	82	4819
WAVEFORM	57	333983	66	18911
PENBASED	96	6625	94	1553
SHUTTLE	96	6592	99	1350
Promedio	**71**	**67183**	**78**	**4140**

Tabla 6.3. Error de clasificación y tasa de reducción usando el clasificador KNN

Conjunto	Sin sel. Inst.		CNN		ProgCNN	
	Error	Reduc.	Error	Reduc.	Error	Reduc.
BUPA	36.26%	0%	39.71%	41	40.58%	62
SEGMENT	4.66%	0%	4.77%	87	5.77%	81
ABALONE	36.26%	0%	43.86%	42	44.16%	61
LANDSAT	8.95%	0%	11.43%	80	12.61%	84
WAVEFORM	23.33%	0%	29.06%	58	28.21%	72
PENBASED	0.66%	0%	1.43%	96	1.48%	93
SHUTTLE	0.17%	0%	0.27%	99	0.14%	99
Promedio	**15.76%**	**0.00%**	**18.65%**	**72**	**18.99%**	**79**

Tabla 6.4. Tasa de reducción y costo computacional usando el clasificador KNN

Conjunto	CNN		ProgCNN	
	Reduc.	CCS	Reduc.	CCS
BUPA	41	60	62	26
SEGMENT	87	1094	81	353
ABALONE	42	30099	61	2068
LANDSAT	80	92332	84	4151
WAVEFORM	58	331804	72	14324
PENBASED	96	6625	93	1762
SHUTTLE	99	7175	99	2180
Promedio	**72**	**67027**	**79**	**3552**

Tabla 6.5. Error de clasificación y tasa de reducción usando el clasificador RPART

Conjunto	Sin sel. Inst.		CNN		ProgCNN	
	Error	Reduc.	Error	Reduc.	Error	Reduc.
BUPA	31.68%	0%	40.14%	41	38.12%	74
SEGMENT	8.17%	0%	24.72%	87	10.91%	80
ABALONE	31.68%	0%	38.71%	41	38.87%	55
LANDSAT	18.66%	0%	21.04%	80	21.24%	83
WAVEFORM	26.63%	0%	26.68%	58	27.70%	69
PENBASED	18.21%	0%	27.39%	96	21.88%	93
SHUTTLE	0.53%	0%	0.65%	96	0.21%	99
Promedio	**19.37%**	**0.00%**	**25.62%**	**71**	**22.70%**	**79**

Tabla 6.6. Tasa de reducción y costo computacional usando el clasificador RPART

Conjunto	CNN		ProgCNN	
	Reduc.	CCS	Reduc.	CCS
BUPA	41	60	74	26
SEGMENT	87	980	80	321
ABALONE	41	30251	55	2639
LANDSAT	80	109928	83	4951
WAVEFORM	58	332596	69	16780
PENBASED	96	10610	93	2135
SHUTTLE	96	7205	99	2465
Promedio	**71**	**70233**	**79**	**4188**

Capítulo VII

Combinación de la detección de ruido y la selección de variables e instancias

7.1. Introducción

En este capítulo se presentan los resultados finales del método de detección de ruido (ver capítulo cinco), integrada a los algoritmos de la selección de variables e instancias, presentadas en los capítulos cuatro y seis, respectivamente. La sección 7.2, se centra en la limpieza de los conjuntos de datos, detectando y eliminando el ruido en las clases usando el algoritmo QcleanNOISE. En la sección 7.3, se combina y analiza el efecto de realizar la detección de ruido y la selección de variables e instancias.

Para realizar este análisis, se consideran los dieciséis conjuntos de datos que se trabajaron en el capítulo dos, estos son: *abalone, balance, breastw, bupa, census, diabetes, ionosfera, iris, landsat, penbased, segment, shuttle, sonar, waveform, letter y vehicle* (ver anexo A). Los clasificadores utilizados para calcular el error de clasificación y realizar las comparaciones de rendimiento son: LDA, KNN y RPART, usando validación cruzada con diez particiones.

7.2. Efecto de la eliminación de ruido en las clases sobre el error de clasificación

La tabla 7.1 muestra los niveles de ruido detectados usando el algoritmo QcleanNOISE. Se observa que los conjuntos con menores niveles de ruido detectados son: *breastw, iris, penbased, segment, shuttle y letter.* Mientras que los conjuntos con mayores niveles de

ruido en las clases son: *vehicle, abalone, waveform, bupa, diabetes y census.* El resto de los conjuntos (*balance, ionosfera, landsat, sonar*) muestran niveles intermedios de ruido (entre 5 y 10%).

Las tablas 7.2 y 7.3 presentan las comparaciones de los errores de clasificación antes y después de eliminar el ruido en los conjuntos de datos que tienen un nivel de ruido mayor al 5% (ver tabla 7.1). En promedio para estos 10 conjuntos de datos se tiene que el error de clasificación disminuye en un 31% para el clasificador LDA, un 54% para el clasificador KNN y un 32% para el clasificador RPART. Para el clasificador LDA, la mayor disminución se da con el conjunto de datos ionosfera (51%), seguido de *vehicle* (43%), *abalone* (41%) y diabetes (41%). En el caso del clasificador KNN, el conjunto de datos con una mayor disminución en promedio para el error de clasificación es *landsat* (71%), le siguen los conjuntos de datos ionosfera (67%), *abalone* (66%) y *vehicle* (63%). Mientras que para el clasificador RPART, la mayor disminución en el error de clasificación se da con los conjuntos de datos *abalone* y *vehicle* con un 47% de reducción, respecto al error de clasificación usando el conjunto de datos sin eliminar el ruido.

De acuerdo con estos resultados, el clasificador KNN, se muestra más sensible a la presencia de ruido en las clases que los clasificadores LDA y RPART, los cuales en promedio presentan la misma sensibilidad. Este mayor efecto del ruido en las clases sobre el error de clasificación usando KNN, también quedo en evidencia cuando se realizaron los estudios experimentales del capítulo 5.

Tabla 7.1. Tasa porcentual de ruido detectada en cada conjunto de datos

Conjunto	%Ruido
ABALONE	26.65
BALANCE	8.32
BREASTW	2.05
BUPA	15.94
CENSUS	11.40
DIABETES	15.23
IONOSFERA	9.97
IRIS	1.33
LANDSAT	6.87
LETTER	5.26
PENBASED	0.59
SEGMENT	3.98
SHUTTLE	1.05
SONAR	8.65
VEHICLE	27.78
WAVEFORM	16.49

Tabla 7.2. Error de clasificación antes y después de eliminar el ruido en cada conjunto de datos

CONJUNTO	Antes de eliminar el ruido			Despues de eliminar el ruido		
	LDA	KNN	RPART	LDA	KNN	RPART
ABALONE	36.12	38.82	37.62	21.21	13.05	19.89
BALANCE	13.28	13.62	21.57	12.20	10.26	17.94
BUPA	32.03	36.26	31.68	24.59	19.14	21.69
CENSUS	17.51	27.38	15.91	13.81	12.79	10.68
DIABETES	22.99	30.48	25.96	13.50	13.92	16.39
IONOSFERA	14.62	15.61	12.39	7.15	5.16	7.15
LANDSAT	16.07	8.95	18.66	11.59	2.63	12.81
SONAR	25.29	18.61	30.05	20.00	11.32	24.47
VEHICLE	22.16	35.01	32.29	12.73	13.09	17.14
WAVEFORM	13.93	23.33	26.63	9.19	11.34	21.89
Promedio	**21.40**	**24.81**	**25.28**	**14.60**	**11.27**	**17.00**

Tabla 7.3. Razón del error de clasificación antes y después de eliminar el ruido en cada conjunto de datos

CONJUNTO	LDA	KNN	RPART
ABALONE	0.59	0.34	0.53
BALANCE	0.92	0.75	0.83
BUPA	0.77	0.53	0.68
CENSUS	0.79	0.47	0.67
DIABETES	0.59	0.46	0.63
IONOSFERA	0.49	0.33	0.58
LANDSAT	0.72	0.29	0.69
SONAR	0.79	0.61	0.81
VEHICLE	0.57	0.37	0.53
WAVEFORM	0.66	0.49	0.82
Promedio	**0.69**	**0.46**	**0.68**

7.3. Efecto de la eliminación de ruido y la selección de variables e instancias sobre el error de clasificación

La tabla 7.4 muestra los errores de clasificación para el clasificador KNN, utilizando los conjuntos de datos, luego de detectar y eliminar el ruido en las clases usando el algoritmo QcleanNOISE. Se observa un incremento en el error de clasificación promedio de aproximadamente 1%. En los conjuntos con una mayor complejidad se observa un incremento en el error de clasificación luego de seleccionar las variables. El error de clasificación para el conjunto de datos abalone, se incrementa en un 9%; para el conjunto de datos bupa en un 8%; para diabetes en un 19% y para vehicle 31%. Mientras que para los conjuntos sonar y waveform, los errores de clasificación disminuyen en un 9% y 12% respectivamente. Los otros conjuntos en la tabla (balance, census, ionosfera), no presentan variación en el error de clasificación, luego de seleccionar las variables.

Tabla 7.4. Comparación del error de clasificación usando KNN después de eliminar el ruido y seleccionar las variables

CONJUNTO	Elim. Solo ruido	Elim. Ruido y sel. Var.	Razón
ABALONE	13.05	14.25	1.09
BALANCE	10.26	10.23	1.00
BUPA	19.14	20.69	1.08
CENSUS	12.79	12.77	1.00
DIABETES	13.92	16.50	1.19
IONOSFERA	5.16	5.09	0.99
LANDSAT	2.63	3.95	1.50
SONAR	11.32	10.26	0.91
VEHICLE	13.09	17.10	1.31
WAVEFORM	11.34	10.02	0.88
Promedio	**11.27**	**12.09**	**1.07**

La tabla 7.5, presenta los errores de clasificación luego de seleccionar las variables e instancias (Error1) y los errores de clasificación después de eliminar el ruido y seleccionar las variables e instancias (Error2). Por otro lado, RED1 representa la reducción global que se obtiene de seleccionar las variables e instancias, mientras que RED2, es la tasa de reducción total luego de eliminar el ruido y seleccionar las variables e instancias.

Tabla 7.5. Comparación del error de clasificación usando KNN antes y después de eliminar el ruido y seleccionar las variables e instancias

CONJUNTO	Error1	RED1	Error2	RED2
ABALONE	38.78	48.75	17.77	78.06
BALANCE	15.84	59.81	12.16	67.53
BUPA	38.84	70.24	25.10	81.86
CENSUS	25.47	67.59	16.45	87..8
DIABETES	29.36	73.29	17.33	85.16
IONOSFERA	15.53	97.00	9.84	98.49
LANDSAT	12.19	88.21	7.83	94.00
SONAR	25.00	81.52	23.84	86.70
VEHICLE	35.28	72.60	20.75	84.43
WAVEFORM	18.22	83.86	11.36	87.80
Promedio	**25.45**	**74.29**	**16.24**	**84.89**

En promedio, el error de clasificación para los conjuntos sin eliminar el ruido en las clases es de 25.45%, con una reducción global promedio de 74.29%. Para los conjuntos en los que se procedió a eliminar el ruido en las clases, el error de clasificación promedio es de 16.24%, con una reducción global del 84.89%.

La detección y eliminación de ruido en las clases trae consigo otro beneficio adicional que consiste en un menor costo computacional, debido a dos razones: la primera, es que al eliminar el ruido en las clases se tendrá un menor número de instancias en el conjunto de entrenamiento y por ende se procesará más rápido. La segunda razón es que un conjunto más “limpio”, permite un mejor rendimiento en algunos de los algoritmos de la minería de

datos debido a que se simplifica la búsqueda y dependiendo del algoritmo se realizan menos iteraciones. La tabla 7.6 muestra la comparación de los costos computacionales en segundos de seleccionar variables e instancias antes y después de eliminar el ruido en las clases.

Tabla 7.6. Comparación del costo computacional en segundos antes y después de eliminar el ruido y seleccionar las variables e instancias

CONJUNTO	CCS1	CCS2
ABALONE	4296.0	973.95
BALANCE	46.6	35.29
BUPA	15.8	8.91
CENSUS	166563.4	65728.30
DIABETES	79.7	35.39
IONOSFERA	8.9	4.27
LANDSAT	23957.3	6061.38
SONAR	5.3	3.68
VEHICLE	130.5	38.86
WAVEFORM	49139.3	23914.00
Promedio	**24424.3**	**9680.40**

Capítulo VIII

Conclusiones y Trabajo Futuro

8.1. Conclusiones

1. Las medidas propuestas para identificar el grado de complejidad de un problema de clasificación asociado a la estructura de los conjuntos de datos, son más eficientes y precisas que las que están disponibles en la literatura. Además, ellas muestran altos niveles de correlación con los errores de clasificación.

2. El método filtro para seleccionar variables que se propone en esta investigación, mostró un mejor rendimiento que el método filtro ReliefF, en términos del error de clasificación y los tiempos de ejecución.

3. La presencia de ruido en las clases produce un deterioro en el rendimiento de los clasificadores. Sin embargo, el grado del efecto de la presencia de ruido varía de un clasificador a otro. Por ejemplo, el clasificador basado en los vecinos más cercanos (KNN), es más sensible a la presencia de ruido en las clases que los clasificadores LDA y RPART.

4. La detección y posterior eliminación del ruido en las clases en el conjunto de entrenamiento de un problema de clasificación supervisada, es un paso importante en el pre-procesamiento de los datos ya que produce una mejora en el nivel de precisión de los clasificadores que hacen uso del conjunto de entrenamiento que ha sido "limpiado".

5. El algoritmo propuesto para la detección de ruido en las clases en el conjunto de entrenamiento (QcleanNOISE), muestra mejores resultados a altos niveles de ruido que los métodos para filtrar el ruido basados en clasificadores . Esto se debe a que los clasificadores son sensibles a la presencia de ruido y por ende tienden a perder precisión a altos niveles de ruido.

6. El método para detectar ruido QcleanNOISE, presenta una mejor eficiencia computacional que los algoritmos basados en clasificadores.

7. El muestreo progresivo permite mejorar la escalabilidad de los algoritmos de selección de instancias.

8. El uso del muestreo progresivo permite mejorar el rendimiento de los algoritmos de selección de instancias al incrementar la tasa de reducción con menores costos computacionales.

8.2. Trabajo Futuro

1. Extender las medidas de complejidad propuestas a conjuntos de datos con valores perdidos.

2. Aplicar las medidas de complejidad propuestas a la selección de instancias, para elaborar algoritmos eficientes que reduzcan el conjunto de entrenamiento en problemas de clasificación supervisada.

3. Aplicar los métodos de selección de variables a conjuntos de datos provenientes de experimentos de microarreglos, en los cuales se tiene una gran cantidad de variables predictoras.

4. Analizar la aplicación y adaptación de las medidas de complejidad a problemas de detección de *outliers* y de clasificación no supervisada.

5. Aplicar métodos ponderados de selección de muestras, usando estimaciones no paramétricas para los pesos de las instancias.

Capítulo IX

Ética

9.1. Introducción

En la actualidad toda propuesta de avance tecnológico está asociada con una serie de implicaciones sociales. El impacto de la misma en la sociedad puede tener efectos positivos y negativos. Hasta hace unas décadas, toda propuesta de una nueva tecnología implicaba un beneficio social, por lo que se veía como sinónimo de progreso y bienestar. Hoy en día, puede estar asociada a efectos negativos, que han motivado que la responsabilidad profesional cobre un papel fundamental. El investigador tiene la obligación ética y moral de evaluar y tomar conciencia sobre las implicaciones que tenga su aporte tecnológico. Un científico debe utilizar toda su capacidad, conocimientos y su juicio para obtener y cumplir cada uno de sus objetivos planteados. La meta final del profesional es ser responsable de la creación de productos tecnológicos útiles y seguros, que no afecten la seguridad, la salud y el bienestar de la sociedad.

9.2. Ética de la investigación

El trabajo del investigador, por ser una actividad encaminada a la búsqueda de un conocimiento de la realidad física, lleva consigo, como exigencia propia, la obligación de veracidad de todas y cada una de las etapas por las que atraviesa la investigación. Desde el planteamiento del problema, que es objeto de estudio, hasta la realización de los experimentos y a la interpretación y comunicación de los resultados que obtiene. Esta exigencia ética tiene su raíz en la naturaleza misma de la actividad científica y requiere que el investigador pueda realizar dicha actividad libremente (Universidad de Navarra, 2002).

Según Routio (2007), los cuatro aspectos que relacionan una investigación y su contexto son:

1. Ética de elegir los problemas a investigar
2. Ética de la recolección y manejo de los datos
3. Ética de la publicación
4. Ética de la aplicación

Ética de elegir los problemas a investigar

El objetivo fundamental de la investigación científica es recolectar conocimiento confiable sobre el mundo. Sin embargo, no es la meta única ni necesariamente suprema en las vidas de la gente común y corriente. Otros objetivos usuales de la vida humana diaria incluyen aspiraciones prácticas innumerables de la vida cotidiana, o hablando en términos más generales: el placer personal, paz, seguridad, gozar de la libertad de la acción y de otros derechos humanos, y para alguna gente la salvación personal del alma. Para lograr cualquiera de éstos, el conocimiento obtenido de una investigación puede ayudar a veces, pero no siempre.

Ética de la recolección y manejo de los datos

Debe ser innecesario precisar que en una investigación científica uno de los comportamientos incorrectos más dañinos es la falsificación de datos o resultados. El daño más grave que se causa no es que el infractor alcance indebidamente un grado académico; lo peor es que la información inventada tal vez vaya a ser usada de buena fe por otros, lo que puede conducir a muchos trabajos infructuosos.

Ética de la publicación

El progreso en la ciencia significa acumulación del conocimiento: las generaciones sucesivas de investigadores construyen su trabajo sobre la base de los resultados alcanzados por científicos anteriores. En este sentido, la Academia Nacional Americana de las Ciencias publicó en 1994 un panfleto titulado "On Being a Scientist - Responsible Conduct in Research" (Ser científico - Conducta responsable en la investigación) (NAS, 1995), donde se tratan las relaciones entre científicos. Aquí se presentan una cita de este libro:

"El principio de justicia y el papel del reconocimiento personal dentro del sistema de retribución de la ciencia explican el énfasis dado a la correcta atribución de los créditos. En el trabajo científico estándar, el crédito se reconoce explícitamente en tres lugares: en la lista de autores, en el reconocimiento de contribuciones de otros y en la lista de referencias o citas. En cualquiera de estos lugares pueden surgir conflictos en torno a la atribución adecuada."

Ética de la aplicación

Durante las fases de iniciación y planeamiento de un proyecto de investigación que ha sido propuesto, el proyecto se examina cuidadosamente desde un ángulo posible, el punto de vista de aquellas personas que se espera que se beneficien del proyecto. Esto es correcto, pero el investigador debe también pensar en las posibles desventajas, especialmente las que pueden derivarse para personas distintas de los creadores del proyecto. Si los resultados de la investigación van a ser aplicados en cierta profesión, por ejemplo dentro de los campos de la enseñanza, la arquitectura o la ingeniería, el investigador quizás podría atender a las normas tradicionales de la profesión a título de orientación. Las asociaciones de algunas profesiones han emitido documentos llamados "Código ético de la profesión", "Código deontológico", etc.

9.2. Ética de la tesis

En la presente tesis, se plantea la generación de una metodología computacional aplicado al reconocimiento de patrones supervisado con la finalidad de ganar eficiencia en la aplicación de los diferentes algoritmos disponibles en la minería de datos, para la extracción de conocimiento en bases de datos. Concientes de la realidad en el campo del desarrollo tecnológico y los fundamentos básicos de ética, se puede afirmar que esta metodología está enmarcada dentro del principio ético de responsabilidad profesional, que será puesto a disposición de la comunidad científica para su mejor aplicación y desarrollo.

Bibliografía

[1] Acuña, E. y Rodríguez, C. (2004). The effect of outliers on the misclassification error rate. Proceedings of the IPSI 2004, Venecia, Italia. Noviembre 10-14, 2004. Disponible en formato CD-ROM.

[2] Aha, D. W., Kibbler, D. y Albert, M. K. (1991). Instance-based learning algorithms. Machine Learning, 6, 37-66.

[3] Attoor, S. N. y Dougherty, E. R. (2004). Classifier performance as a function of distributional complexity. Pattern Recognition, 37. 1641-1651.

[4] Bernadó, E. y Ho, T. K. (2004). On classifier domain of competence. Proceedings 17th. Int. Conference on Pattern Recognition, Cambridge, UK, pp.136-139.

[5] Bernadó, E. y Ho, T. K. (2005). Domain of competence of XCS classifier system in complexity measurement space. IEEE Transactions on Evolutionary Computation, 9, 82-104.

[6] Bernadó, E., Ho, T. K. y Orriols, A. (2006). Data complexity and evolutionary learning: Classifier's behavior and domain of competence. In: Ho, T. K. y Basu M. (Eds.) Data Complexity in Pattern Recognition, Springer, London, pp. 115-134.

[7] Blake, C. L. y Merz, C. J. (1998). UCI: repository of machine learning databases. *URL http://www. ics. uci. edu/~ mlearn/MLRepository. html.*

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

[9] Blum, A. L. y Rivest, R. L. (1992) Training a 3-node neural network is NP-complete, Neural Networks, 5, 117-127.

[10] Breiman, L. (1996). Bagging predictors. Machine Learning, 24, 123-140.

[11] Breiman, L. (1998). Arcing classifiers. Annals of Statistics, 26, 801-824.

[12] Breiman, L. (1999). Prediction games & arcing algorithms. Neural Computation, 11, 1493-1517.

[13] Breiman, L., Friedman, J., Olshen, R. A., y Stone, C. J. (1984). Classification and Regression Trees. Califirnia, USA: Wadsworth, Inc.

[14] Brighton, H. y Mellish, C. (2002). Advances in Instance Selection for Instance-Based Learning Algorithms, Data Mining and Knowledge Discovery, 6, 153-172.

[15] Brodley, C. E. y Friedl, M. A. (1996). Identifying and eliminating mislabeled training instances, Proceedings of 13th National Conf. on Artificial Intelligence, pp.799-805.

[16] Brodley, C. E. y Friedl, M. A. (1999). Identifying mislabeled training data, Journal of Artificial Intelligence Research, 11, 131-167.

[17] Cano, J. R., Herrera, F. y Lozano, M. (2003). Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study. IEEE Transactions on Evolutionary Computation, 7, 561-575.

[18] Cano, J. R., Herrera, F. y Lozano, M. (2005). Stratification for scaling up evolutionary prototype selection. Pattern Recognition Letters, 26, 953-963.

[19] Cochran, W. (1977). Sampling Techniques. John Wiley & Sons. New York.

[20] Cover, T. y Hart, P. (1967). Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory, 13, 21-27.

[21] Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. Proceedings of the Eighteenth International Conference on Machine Learning, pp. 74-81.

[22] Dash, M. y Liu, H. (1997). Feature selection for classifications. Intelligent Data Analysis: An International Journal, 1, 131-156.

[23] Devijver, P. A. y Kittler, J. (1982). Pattern Recognition. A Statistical Approach. Prentice-Hall, Englewood Cliffs, London.

[24] Domingo, C., Gavalda, R., y Watanabe, O. (2002). Adaptive sampling methods for scaling up knowledge discovery algorithms. Data Mining and Knowledge Discovery, An International Journal, 6,131-152.

[25] Domingos, P. y Hulten, G. (2002). Learning from infinite data in finite time. Advances in Neural Information Processing Systems, 14, 673-680.

[26] Duda, R., Hart, P. y Stork, D. (2000). Pattern Classification (2nd Edition). Wiley Interscience. New York.

[27] Fayyad, U. M., Piatetsky-Shapiro, G., y Smyth, P. (1996). From data mining to knowledge discovery: an overview. Advances in Knowledge Discovery and Data Mining, pp. 495-515. The MIT Press, Menlo Park, CA.

[28] Frawley, W. J., Piatetsky-Shapiro, G. and Matheus, C. J. (1991). Knowledge Discovery in Databases: An Overview, In: Knowledge Discovery in Databases (G. Piatetsky-Shapiro y C. J. Matheus, eds.), AAAI Press / MIT Press, Menlo Park, CA.

[29] Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. The Annals of Statistics. 19, 1-67.

[30] Friedman, J. H. y Rafsky L. C. (1979). Multivariate Generalizations of the Wald-Wolfowitz and Smirnov Two Sample Tests, The Annals of Statistics, 7, 697-717.

[31] Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition. Academic Press. San Francisco, CA.

[32] Fukunaga, K. y Hayes, R. (1989). Effects of sample size in classifier design. IEEE Transactions on Pattern Anal. Machine Intell., 8, 873-885.

[33] Gamberger, D., Lavrac, N. and Dzerobski, S. (2000). Noise detection and elimination in data preprocessing: Experiments in medical domains. Applied Artificial Intelligence.

[34] Gamberger, D., Lavrac, N. and Groselj, C. (1999). Experiments with noise filtering in a medical domain, Proc. of 16th ICML Conference, pp. 143-151, San Francisco, CA.

[35] Gates, G. W. (1972). The Reduced Nearest Neighbor Rule. IEEE Transactions on Information Theory, Vol. IT-18(3): 431-433.

[36] Gavaldá, R. y Watanabe, O. (2001). Sequential sampling algorithms: Unified analysis and lower bounds, Proc. 1st International Symposium on Stochastic Algorithms, LNCS 2264, pp. 173-187.

[37] Gu, B., Liu, B., Hu, F. y Liu, H. (2001). Efficiently Determining the Starting Sample Size for Progressive Sampling. In Workshop Notes of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.

[38] Guyon, I., Matic, N. y Vapnik, V. (1996). Discovering informative patterns and data cleaning. Advances in Knowledge Discovery and Data Mining, 181-203. AAAI/MIT Press, Menlo Park, CA.

[39] Han, J. y Kamber, M. (2006). Data Mining: Concepts and Techniques, 2nd ed. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, San Francisco, CA.

[40] Hand D. J. (1997). Construction and Assessment of Classification Rules. John Wiley and Sons. Chichester, UK.

[41] Hart, P. E. (1968). The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory, 14, 515-516.

[42] Hernández J., Ramirez, M. J. y Ferry, C. (2004). Introducción a la minería de datos. Pearson.

[43] Ho, T. K. (2001). Data Complexity Analysis for Classifier Combination. Proceedings of the Second International Workshop on Multiple Classifier Systems, pp. 53- 67.

[44] Ho, T. K. (2002). A data complexity analysis of comparative advantages of decision forest constructors, Pattern Analysis and Applications, 5, 102-112.

[45] Ho, T. K. y Basu, M. (2000). Measuring the complexity of classification problems, Proceedings. 15th International Conference on Pattern Recognition, 2, 43- 47.

[46] Ho, T. K. y Basu, M. (2002). Complexity measures of supervised classification problems, IEEE Trans. on Pattern Analysis and Machine Intelligence, 24, 289-300.

[47] Ho, T. K. y Bernadó, E. (2006). Data complexity and domains of competence of classifiers. In: Ho, T. K. y Basu M. (Eds.) Data Complexity in Pattern Recognition, Springer. pp. 135-152.

[48] Hughes, G. F. (1968). On the mean accuracy of statistical pattern recognizers. IEEE Transactions on Information Theory, 14, 55-63.

[49] Jain, A. K., Duin, R. P. W., y Mao, J. (2000). Statistical Pattern Recognition: A Review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22, 4-37.

[50] Jain, A. K., Xu, X., Ho, T. K., y Xiao, F. (2002). Uniformity Testing Using Minimal Spanning Tree. Proceedings of the 16th International Conference on Pattern Recognition, pp. 281-284.

[51] John, G. H. (1995). Robust decision trees: Removing outliers from databases. Proc. of the First International Conference on Knowledge Discovery and Data Mining, pp.174-179, AAAI Press, Menlo Park, CA.

[52] John, G. y Langley, P. (1996). Static versus dynamic sampling for data mining. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, August, 2-4, Portland, OR, E. Simoudis, J. Han, y U. Fayyad (Eds.). AAAI Press, Menlo Park, CA., pp. 367-370.

[53] Kantardzic, M. (2001). Data Mining Concepts Models Methods and Algorithms. Wiley Interscience. John Wiley & Sons, Inc. New York.

[54] Kibbler D. y Aha D. W. (1987). Learning representative exemplars of concepts: An initial case of study. Proceedings of the Fourth International Workshop on Machine Learning, pp. 24-30.

[55] Kim, W., Choi, B., Hong, E. y Lee, D. (2003). A taxonomy of dirty data. Data Mining and Knowledge Discovery, 7, 81-89.

[56] King, R. D., Feng, C. y Sutherland, A. (1995). STATLOG: comparison of classification algorithms on large real-world problems. Applied Artificial Intelligence, 9, 289-334.

[57] Kira, K. y Rendall, L. A. (1992). A practical approach to feature selection, Proceedings of the Ninth International Conference on Machine Learning, Aberdeen, Scotland, UK, Morgan Kaufmann Publishers, San Mateo, pp. 249–256.

[58] Kivinen, J. y Mannila, H. (1994). The power of sampling in knowledge discovery. Proceedings of PODS'94, pp.77-85.

[59] Kohavi, R. y John, G. (1997). Wrappers for feature subset selection. Artificial Intelligence, 97, 273-324.

[60] Kononenko, I. (1994). Estimating attributes: Analysis and extension of RELIEF. In: F. Bergadano and L. De Raedt, editors, Proceedings of the European Conference on Machine Learning, pp. 171-182, Catania, Italy. Berlin: Springer-Verlag.

[61] Lawrence, N. D. y Schölkopf, B. (2001). Estimating a kernel fisher discriminant in the presence of label noise. Proceedings of the eighteen International Conference on Machine Learning , pp 306-313.

[62] Lee, Y. y Hwang, K.-W. (1996). Selecting good speech features for recognition, ETRI Journal, 18, 29-41.

[63] Li, X. (2002). Data Reduction via Adaptive Sampling. Communications in Information and System, 2, 53-58.

[64] Li, Y. (2004). Classification on the presence of class Noise. M.Sc. Thesis, pp. 30, Delft University of Technology.

[65] Liu, H. y Motoda, H. (1998). Feature Selection for Knowledge Discovery and Data Mining. Boston: Kluwer Academic Publishers.

[66] Liu, H. y Motoda, H. (editores). (2002). Instance Selection and Construction for Data Mining, Kluwer Academic, Norwell, MA.

[67] Liu, H. y Setiono, R. (1996). Feature selection and classification - a probabilistic wrapper approach. In Proc. 9th International Conference on Industrial & Engineering Applications of AI and Expert Systems, Fukuoka, Japan, pp. 419-424.

[68] Liu, H. y Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering, IEEE Transactions on Knowledge and Data Engineering, 17, 491-502

[69] Liu, H., Motoda, H. y Yu, L. (2002). Feature selection with selective sampling. Proceedings of the Nineteenth International Conference on Machine Learning, pp. 395-402.

[70] Liu, H., Yu, L., Dash, M. y Motoda, H. (2003). Active feature selection using classes. Proceedings of the Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-03) pp. 474-485.

[71] McLachlan, G. (1992). Discriminant analysis and statistical pattern recognition. John Wiley & Sons. New York.

[72] Michie, D. Spiegelhalter, D.J. y Taylor, C.C. editors. (1994). Machine Learning, Neural and Statistical Classification. Ellis Horwood.

[73] Mollineda, R. A., Sanchez, J. S. y Sotoca, J. M. (2005). Data characterization for effective prototype selection. Proceedings 2nd. Iberian Conference on Pattern Recognition and Image Analysis, Estoril, Portugal, pp. 27-34.

[74] Mount, D. M. y Arya, S. (1997). ANN: A library for approximate nearest neighbor searching. CGC 2nd Annual Fall Workshop on Computational Geometry, URL: http://www.cs.umd.edu/~mount/ANN

[75] NAS (1995). On Being A Scientist: Responsible Conduct In Research. Consultado el 18 de mayo, 2007. Disponible en la página Web: http://www.nap.edu/readingroom/books/obas/

[76] Oates, T. y Jensen, D. (1998). Large Datasets Lead to Overly Complex Models: An Explanation and a Solution, Proc. Fourth Int'l Conf. on Knowledge Discovery and Data Mining, pp 294-298.

[77] Provost, E., Jensen, D. y Oates, T. (1999). Efficient progressive sampling. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August, 15-18, San Diego, CA, USA, S. Chaudhuri y D. Madigan (Eds.). New York: ACM, pp. 23-32.

[78] Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1, 81-106.

[79] Quinlan, J. R. (1993). C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA.

[80] R Development Core Team (2006). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

[81] Ritter, G. L., Woodruff, H.B., Lowry, S.R. and Isenhour, T.L. (1975). An Algorithm for a Selective Nearest Neighbor Decision Rule. IEEE Transactions on Information Theory, 21, 665-669.

[82] Routio P. (2007). Ética de la investigación. Consultado el 3 de mayo, 2007. Disponible en: http://www2.uiah.fi/projects/metodi/251.htm

[83] Singh, S. (2003a). PRISM - A novel framework for pattern recognition, Pattern Analysis and Applications, 6,134-149.

[84] Singh, S. (2003b). Multiresolution estimates of classification complexity. Pattern Analysis and Machine Intelligence, IEEE Transactions, 25, 1534 – 1539.

[85] Smith, F. W. (1968). Pattern classifier design by linear programming, IEEE Trans. on Computers, 17, 367-372.

[86] Sohn, S. Y. (1999). Meta analysis of classification algorithms for pattern recognition, IEEE Trans. on Pattern Analysis and Machine Intelligence, 21, 1137-1144.

[87] Sotoca, J. M., Sanchez, J. S. y Mollineda, R. A. (2005). A review of data complexity measures and their applicability to pattern classification problems. Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA, pp. 77-83.

[88] Stanfill, C. y Waltz, D. (1986). Toward memory-based reasoning. Communications of the ACM, 1213–1228.

[89] Stolfo, S., Prodromidis, A., Tselepis, S., Lee, W., Fan, D. and Chan, P. (1997). JAM: Java Agents for Meta-learning over Distributed Databases, Proc. Third Int'l Conf. on Knowledge Discovery and Data Mining, pp. 74-81.

[90] Toivonen, H. (1996). Sampling large databases for association rules. Proceedings of the 22nd International Conference on Very Large Databases, pp. 134-145.

[91] Tomek, I. (1976). An Experiment with the Edited Nearest-Neighbor Rule. IEEE Transactions on Systems, Man, and Cybernetics, 6, 448-452.

[92] Universidad de Navarra (2002). Ética de la investigación científica. Consultado el 2 de mayo, 2007. Disponible en: http://www.unav.es/cdb/medicinainv.html

[93] Vapnik, V. y Lerner, A. (1963). Pattern recognition using generalized portrait method. Automation and Remote Control, 24, 774—780.

[94] Venables, W. N. y Ripley, B. D. (2002). Modern Applied Statistics with S. Fourth Edition. Springer, New York.

[95] Vucetic, S. y Obradovic, Z. (2000). Performance Controlled Data Reduction for Knowledge Discovery in Distributed Databases. Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications, pp: 29-39.

[96] Wang, M., Iyer, B. y Vitter, J. S. (1998). Scalable mining for classification rules in relational databases, In: Proceedings of IDEAS'98, pp. 58-67.

[97] Watanabe, O. (2000). Simple sampling techniques for discovery science. IEICE Trans. Info. Systems, E83-D, 1, pp. 19-26.

[98] Webb, A. (1999). Statistical pattern recognition. Oxford University Press Inc. New York.

[99] Wilson, D. L. (1972). Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Cybernetics, 2, 408-421.

[100] Wilson, D. R. y Martinez, T. R. (1997). Improved Heterogeneous Distance Functions, Journal of Artificial Intelligence Research, 6, 1-34.

[101] Wilson, D. R. y Martinez, T. R. (2000). Reduction Techniques for Instance Based Learning Algorithms. Machine Learning, 383, pp. 257-286, Kluwer Academic Publishers, Boston.

[102] Wilson, W. (1998). Generalization in the XCS Classifier System, Genetic Programming: Proceedings of the Third Annual Conference.

[103] Xing, E., Jordan, M. y Karp, R. (2001). Feature selection for high-dimensional genomic microarray data. Proceedings of the Eighteenth International Conference on Machine Learning, pp. 601-608.

[104] Xiong, H., Pandey, G., Steinbach, M. y Kumar, V. (2006). Enhancing data analysis with noise removal. IEEE Transactions on Knowledge and Data Engineering, 18, 304-319.

[105] Yang, Y. y Pederson, J. (1997). A comparative study on feature selection in text categorization. Proceedings of the 14th International Conference on Machine Learning, pp. 412–420.

[106] Yu, L. y Liu, H. (2003). Efficiently handling feature redundancy in high-dimensional data. Conference on Knowledge Discovery in Data. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. Washington, D.C. KDD, pp. 685-690.

[107] Zeng, X. y Martinez, T. (2003). A noise filtering method using neural networks. SCIMA 2003. IEEE International Workshop on Soft Computing Techniques in Instrumentation, Measurement and Related Applications, 17 May 2003, pp. 26–31.

[108] Zhu, X. y Wu, X. (2006). Scalable representative instance selection and ranking. Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006), Hong Kong, pp. 352-355.

[109] Zhu, X., Wu, X. y Chen, Q. (2003). Eliminating class noise in large datasets. Proceedings of the 20th ICML International Conference on Machine Learning (ICML 2003). Washington D.C., pp. 920-927.

[110] Zhu, X., Wu, X. y Chen, Q. (2006). Bridging Local and Global Data Cleansing: Identifying Class Noise in Large, Distributed Data Datasets. *Data Mining and Knowledge Discovery*, 12, 275-308.

Anexo A. Conjuntos de datos

Para realizar las evaluaciones experimentales en esta tesis se utilizan diez y seis bases de datos. Estas bases de datos se presentan en la tabla A1, conjuntamente con el número de instancias, el número de clases y el número de variables y las distribución del número de instancias por clase. Estos conjuntos de datos están a disposición en "*The repository of Machine Learning Databases*" el cual es mantenido por el Departamento de Ciencias de Computadora de la Universidad de California en Irvine (Blake y Merz, 1998). A continuación se presenta una breve descripción de cada una de las base de datos.

Abalone. Los datos de este conjunto se usan para predecir la edad del *abalone* usando medidas físicas. La edad del *abalone* es determinada cortando su cararazón a través del cono, manchándola, y contando el número de los anillos a través de un microscopio. Este conjunto de datos contiene 4177 instancias, con 8 variables numéricas y tres clases.

Balance. Estos datos fueron generados para modelar resultados experimentales psicológicos. El conjunto de datos contiene 625 instancias, con 4 variables numéricas y tres clases

Breastw. Este conjunto de datos proviene del hospital de la universidad de *Wisconsin*. Contiene una lista de casos clínicos sobre tumores cancerigenos, recolectados durante el período 1989-1991. Este conjunto de datos contiene 699 instancias; cada instancia tiene 9 variables continuas y una variable categórica que identifica a las dos clases (benigno y maligno). En esta tesis se usa únicamente 683 instancias, debido a que se descartan 16 instancias que contienen valores perdidos.

Bupa. Este conjunto de datos sobre desordenes del hígado, generado por la fundación para investigaciones médicas BUPA, contiene 345 instancias, con 7 variables continuas y dos clases.

Census. Este conjunto de datos contiene información demográfica del censo de los Estados Unidos en el año 1990. Contiene 32561 instancias, con 13 variables (6 variables continuas y 7 nominales). Cada instancia se clasifica en una de dos clases: "ingresos < 50000" o "ingresos $\geq$50000". Un 7% de las instancias tiene información incompleta y han sido eliminados.

Diabetes. Este conjunto de datos también es conocido como *Pima Indians Diabetes*. Contiene 768 instancias, con 8 variables numéricas y dos clases (si tiene o no tiene diabetes).

Ionósfera. El conjunto de datos original contiene a 34 variables, pero se ha eliminado las dos primeras variables, porque la primera característica tiene el mismo valor en una de las clases y la segunda variables tiene el valor de cero para todas las instancias. Este conjunto de datos tiene 351 instancias y dos clases.

Iris. Este conjunto de datos contiene 150 instancias con 4 variables continuas. Tiene 3 clases cada una de las cuales se refiere a una variedad de iris: Iris *setosa*, Iris *versicolor* e Iris *virginica*.

Landsat. Esta base de datos consiste de valores multi espectrales de una imagen obtenida por satélite. El conjunto de datos contiene 6435 instancias, en esta investigación se unió el conjunto de entrenamiento (4435 instancias), con el conjunto de prueba (2000 instancias). Cada instancia representa una imagen de satélite con 36 variables continuas. Los datos se clasifican en seis grupos.

Letter. Este conjunto de datos contiene 17 variables en total: 1 variable categórica y 16 variables numéricas. El número de clases en este conjunto es de 26, que representan a las letras mayúsculas del alfabeto inglés (de A a la Z). El número total de instancias es de 20000.

Penbased. Este conjunto de datos esta relacionado con la identificación de escrituras hechas a mano. Contiene 16 variables numéricas y una variables categórica representando a las 10 clases. El número total de instancias es de 10992

Segment. Contiene 2310 instancias en total con 19 variables continuas y siete clases que representan a 7 diferentes colores, en la segmentación de imágenes. Fue creada por el grupo *Vision* de la Universidad de Massachussets.

Shuttle. El número de instancias para este conjunto de datos es de 58000, con 9 variables continuas y una variable categórica representando a las clases que tiene 7 niveles: Rad.Flow, Fpv.Close, que Fpv.Open, Alto, Desvían, Bpv.Close, Bpv.Open. En este conjunto de datos, aproximadamente el 80% de los datos pertenecen a la clases 1.

Sonar. En este conjunto de datos se tienen 208 instancias en total con 61 variables. Las primeras 60 variables representan la energía dentro de una banda integrada de frecuencia en cierto periodo de tiempo. La última columna representa a las dos clases: 1 si el objeto es una piedra, y 2 si el objeto es un el cilindro de metal. El valor del rango de cada atributo varía de 0.0 a 1.0.

Vehicle. Este conjunto de datos proviene del instituto de Turing en Glasgow, Escocia. El propósito es el de encontrar un método que distinga a los objetos en 3D dentro de una imagen en 2D reconociendo las siluetas de los objetos. El conjunto de datos tiene 846 instancias, con 18 variables continuas y cuatro clases.

Waveform. Conjunto de datos disponible en el repositorio de datos de la UCI que ha sido generado artificialmente. Consiste de 5000 instancias, con 40 variables continuas y tres clases. Cada clase tiene aproximadamente el 33.33% de los datos.

Tabla A. Conjuntos de datos

Conjunto de datos	Número de instancias	Variables	Clases	Instancias por clase
ABALONE	4177	8	3	1407 / 1323 / 1447
BALANCE	625	4	4	288 / 288 / 49
BREASTW	683	9	2	444 / 239
BUPA	345	7	2	145 / 200
DIABETES	768	8	2	500 / 268
IONÓSFERA	351	32	2	225 / 126
IRIS	150	4	3	50 / 50 / 50
LANDSAT	6435	36	6	1533 / 703 / 1358 / 626 / 707 / 1508
LETTER	20000	17	26	789 766 736 805 768 775 773 734 755 747 739 761 792 783 753 803 783 758 748 796 813 764 752 787 786 734
PENBASED	10992	16	10	1143 1144 1055 1144 1055 1056 1142 1055 1055 1143
SEGMENT	2310	16	7	330 / 330 / 330 / 330 / 330 / 330 / 330
SHUTTLE	58000	9	7	45586 / 50 / 171 / 8903 / 3267 / 10 / 13
SONAR	208	60	2	111 / 97
VEHICLE	846	18	4	218 / 212 / 217 / 199
WAWEFORM	5000	40	3	1653 / 1655 / 1692
CENSUS	32561	14(6C, 7N)	2	22654 / 7508