CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL
DEPARTAMENTO DE COMPUTACIÓN

Clasificación de grandes conjuntos de datos vía Máquinas de Vectores Soporte y aplicaciones en sistemas biológicos

Tesis que presenta:

Jair Cervantes Canales

Para obtener el grado de

Doctor en Ciencias

En la especialidad de
Ingeniería Eléctrica

Opción:
Computación

Directores de Tesis:
Dra. Xiaoou Li Zhang
Dr. Wen Yu Liu

México D.F. 18 Agosto 2009

A mis padres y hermanos

A Raquel y Benito

Agradecimientos

Quiero expresar mi agradecimiento a mis asesores Dra. Xiaoou Li y Dr. Wen Yu por todo su apoyo, confianza y guía a lo largo de la tesis.

A los revisores que me ayudaron enormemente con sus comentarios para mejorar este trabajo Dr. Debrup Chakraborty, Dr. Miguel González, Dr. Ivan Lopez y Dr. Luis Rocha.

A mis compañeros y amigos del CINVESTAV: Arturo, Erasmo, Farid, Francisco Javier, Héctor, Israel, Luis Omar, Luis, Juan, Roberto.

Un especial agradecimiento a Sofy, Felipa y Flor por su ayuda administrativa.

Agradezco su apoyo a CONACyT por la beca proporcionada durante mi estancia en el programa de doctorado en el CINVESTAV.

Abstract

Support Vector Machines (SVM) has demonstrated highly competitive performance in many real-world applications. However, despite its good theoretical foundations and generalization performance, SVM is not suitable for large data sets classification, because training kernel matrix grows in quadratic form with the size of the data set, which provokes that training of SVM on large data sets is a very slow process. In the literature, most fast SVM implementations use completely the input data set in order to obtain good classification accuracy. However, the principal disadvantage of SVM is due its excessive computational cost in large data sets. Some algorithms have been proposed in the literature, but the training time is still very large and prohibitive in large data sets.

In this thesis, we present different algorithms that tackle this problem. The proposed algorithms reduce the training time considerably, recovering the most important data points of entire data set and eliminating data points that are not important. The proposed algorithms use two stages of SVM. A first stage on a small data set in order to obtain a sketch of support vectors (SVs) distribution and recovery data points with most chance to be SVs, and a second stage of SVM in order to improve the first classification hyperplane obtained.

Our research is divided into three main parts. First, three algorithms for large data set classification are introduced, the main novelty of the proposed approaches consists in using clustering and sectioning techniques in order to reduce the SVM training time. Second, a multiclassification algorithm is proposed, the main novelty of the approach proposed with respect first three techniques consists on the implementation of a method that search support vectors candidate between the space of support vectors with different label obtained in the first stage, the algorithm reduce even more the support vectors candidate, because the search of support vectors candidate is restricted to a small space. Finally, a SVM algorithm for classification of introns and exons is proposed. Classification of gene sequence into regions that code for genetic material and regions that do not is a challenging task in DNA sequence analysis. Due to enormous quantities of DNA sequence and to the fact that regions that encode

in proteins (exons) can be interrupted by regions that do not encode (introns), recognition of genes is not an easy challenge

The proposed algorithms overcome the main limitation of SVM without affecting in a significant way the quality of results. Moreover, experimental results show that the accuracy obtained by the proposed approach is comparable with other fast SVM implementations. Furthermore, results indicate that our approach is a viable alternative since it improves the training time of the best fast SVM implementations known to date.

Resumen

Las Máquinas de Soporte Vectorial Vectores Soporte (SVM) han demostrado tener un gran desempeño en muchas aplicaciones del mundo real. Sin embargo, a pesar de sus buenos fundamentos teóricos y buen desempeño al generalizar, las SVM no son adecuadas para clasificación con grandes conjuntos de datos, ya que la matriz del kernel crece de forma cuadrática con el tamaño del conjunto de datos, provocando que el entrenamiento de las SVM sobre conjuntos de datos grandes sea un proceso muy lento. En la literatura, la mayoría de los métodos de agilización de SVM usan el conjunto de datos entero con el objetivo de obtener buenas precisiones de clasificación. Sin embargo, la principal desventaja de las SVM es debido a su excesivo costo computacional en conjuntos de datos grandes. Algunos algoritmos han sido propuestos en la literatura pero el tiempo de entrenamiento sigue siendo muy grande y prohibitivo en conjuntos de datos grandes.

En esta tesis, presentamos varios algoritmos que enfrentan este problema. Los algoritmos propuestos reducen considerablemente el tiempo de entrenamiento recuperando los datos más importantes del conjunto de datos entero y eliminando aquellos que no son importantes. Los algoritmos propuestos usan dos etapas de SVM. Una primera etapa sobre un conjunto de datos pequeño con el objetivo de obtener un esbozo de la distribución de los vectores soporte (VS) y recuperar los datos con más probabilidades de ser VS y una segunda etapa de SVM con el objetivo de mejorar el primer hiperplano de clasificación obtenido.

Nuestra investigación está dividida en tres partes principales. Primero, tres algoritmos para clasificación de grandes conjuntos de datos son presentados, la principal novedad de los enfoques propuestos consiste en emplear agrupamiento y/o técnicas de seccionamiento con el objetivo de reducir el tiempo de entrenamiento de las SVM. Segundo, un algoritmo de multiclasificación es propuesto, la principal novedad del algoritmo propuesto con respecto a las tres primeras técnicas consiste en la implementación de un método de búsqueda de candidatos a SVs entre el espacio de los SVs con diferente etiqueta obtenidos en la primera etapa. El algoritmo reduce aún más el conjunto de candidatos a vectores soporte debido a que la búsqueda de candidatos a VS es restringida a un pequeño espacio. Finalmente, un algoritmo para clasificación de Introns y Exones es propuesto. La clasificación de secuencias de genes en regiones que codifican en material genético y regiones que no, es un reto en análisis de

secuencias de ADN. Debido a que enormes cantidades de secuencias de ADN y al hecho de que las regiones que codifican en proteínas (exones) pueden ser interrumpidas por regiones que no codifican (intrones), el reconocimiento de genes no es un reto fácil

Los algoritmos propuestos superan la principal limitación de las SVM sin afectar de forma significativa la calidad de los resultados, además los resultados experimentales muestran que la precisión obtenida mediante los enfoques propuestos es comparable con otras implementaciones de SVM. Además, los resultados indican que los enfoques propuestos son una alternativa viable ya que estos mejoran el tiempo de entrenamiento de las mejores implementaciones de SVM conocidas a la fecha.

Índice general

Índice de Figuras

Índice de Tablas

Índice de Abreviaciones

SVM	Máquinas de Soporte Vectorial
SSVM	Máquinas de Soporte Vectorial Simples
SMO	Optimización Mínima Secuencial
FCM	Fuzzy C-Means
ADN	Ácido Desoxirribonucleico
ARN	Ácido Ribonucleico
MEB	Esferas de Cerradura Mínima
DAGSVM	SVM basadas en el Grafo Acíclico Dirigido
OVA	Estrategia uno contra todos
OVO	Estrategia uno contra uno
FCM-SVM²	Algoritmo SVM en dos etapas basado en FCM
MEB-SVM²	Algoritmo SVM en dos etapas basado en MEB
mMEB-SVM²	Algoritmo SVM multiclase en dos etapas
Bio-SVM²	Algoritmo SVM para clasificación de exones
RS-SVM²	Algoritmo SVM en dos etapas basado en Selección aleatoria

Capítulo 1

Introducción

La revolución digital ha hecho posible que la captura de datos sea fácil y su almacenamiento tenga un costo muy bajo. Como resultado, enormes cantidades de información con diferentes tipos de datos son almacenados continuamente en bases de datos, para un posterior uso y análisis. Esto ha provocado que los métodos habituales empleados en el análisis de datos sean obsoletos al ser empleados en grandes conjuntos de datos. Debido a esto, métodos semiautomáticos de análisis son necesarios, teniendo como objetivo principal entender y analizar enormes cantidades de datos.

La información almacenada en bases de datos crece significativamente día con día. Debido al incremento en la cantidad de datos y aún en las características asociadas a estos datos, se necesitan técnicas de exploración versátiles, robustas y eficientes que puedan hacer frente a los cambios actuales. Estas técnicas de exploración pueden ser supervisadas o no supervisadas. La clasificación de datos es descrita como una técnica de aprendizaje supervisado y un modelo de clasificación puede servir como una herramienta para distinguir entre objetos de diferentes clases [86]. Un proceso de clasificación incluye dos fases: entrenamiento y prueba. En la fase de entrenamiento, un conjunto de datos inicial es usado para decidir que parámetros deberán ser ponderados y combinados con el objetivo de separar varias clases de objetos. El aprendizaje intenta descubrir una representación óptima a partir del conjunto de datos cuya membresía o etiqueta de clase

es conocida. En la fase de prueba, los pesos determinados en la fase de entrenamiento son aplicados a un conjunto de objetos (conjunto de prueba) cuyas etiquetas de clase se desconoce, con el objetivo de determinar su clase. Algunos métodos de clasificación involucran un enfoque heurístico que pretende encontrar la "mejor" solución para el problema de optimización.

Existen varias técnicas de clasificación convencionales en la literatura, e.g. reglas basadas en clasificadores de vecinos cercanos (*nearest neighbor*), clasificadores Bayesianos, redes neuronales artificiales, árboles de decisión y SVM. De las técnicas anteriores, las redes neuronales son una de las técnicas más usadas [61]. Como aproximador universal, las redes neuronales han sido ampliamente utilizadas en un gran número de aplicaciones. Sin embargo, deben tomarse en cuenta muchos factores al construir una red para un problema dado: el algoritmo de aprendizaje, la arquitectura, el número de neuronas por capa, el número de capas, la representación de los datos y mucho más. Además, éstas son muy sensibles a la presencia de ruido en los datos de entrenamiento. Los árboles de decisión, también han sido ampliamente usados en problemas de clasificación. Estos son usualmente más veloces que las redes neuronales en la fase de entrenamiento, Sin embargo, no presentan flexibilidad al modelar los parámetros [76]. Un simple clasificador puede ser el enfoque de vecino cercano [31]. Los métodos de vecino cercano tienen la ventaja de que son fáciles de implementar, no obstante, éstos suelen ser bastante lentos si el conjunto de datos de entrada es muy grande. Por otro lado, estos son muy sensibles a la presencia de parámetros irrelevantes.

De estas técnicas, las SVM es una de las técnicas más conocidas para optimizar la solución esperada [15]. Se ha mostrado que las SVM son superiores a otros métodos de aprendizaje supervisado [20] [63] [73] [80] [85]. Debido a sus buenos fundamentos teóricos y su buena capacidad de generalización las SVM han llegado a ser en los últimos años uno de los métodos de clasificación más utilizado. Las cotas de decisión son determinadas directamente a partir de los datos de entrenamiento al emplear SVM de tal forma que la separación existente (margen) entre las cotas de decisión sea maximizada en un espacio altamente dimensional llamado espacio de características. Esta estrategia de

clasificación minimiza los errores de clasificación de los datos de entrenamiento y obtiene una mejor habilidad de generalización, i.e. las habilidades de clasificación de las SVM y otras técnicas difieren significativamente, especialmente cuando el número de datos de entrada es pequeño. Las SVM son una poderosa técnica empleada en clasificación de datos y análisis de regresión.

Una ventaja notable de las SVM radica en el hecho de que éstas obtienen un subconjunto de vectores soporte durante la fase de aprendizaje, que a menudo es sólo una pequeña parte del conjunto de datos original. Este conjunto de vectores soporte representa una tarea de clasificación dada y es formado por un conjunto compacto de datos. Sin embargo, para encontrar un hiperplano de separación las SVM necesitan resolver un problema de programación cuadrática (QP), que involucra una matriz de densidad $N \times N$, donde N es el número de puntos en el conjunto de datos. Ya que la mayoría de las rutinas de QP tienen complejidad cuadrática, las SVM requieren grandes cantidades de tiempo computacional y memoria para bases de datos muy grandes [67][99], i.e. la complejidad del entrenamiento de las SVM es altamente dependiente del tamaño del conjunto de datos.

En esta tesis se presentan algoritmos de clasificación basados en agrupamiento y seccionamiento de los datos de entrada, concretamente en agrupamiento difuso utilizando *Fuzzy C-Means* (FCM-SVM^2), seccionamiento utilizando Esferas de Cerradura Mínima (MEB-SVM^2) y selección aleatoria (RS-SVM^2), los algoritmos propuestos realizan un reescaneo de datos en los puntos que son vectores soporte con el propósito de refinar y mejorar la precisión de clasificación. En los algoritmos propuestos en lugar de procesar todos los datos, los algoritmos propuestos seccionan o dividen el conjunto de datos de entrada dentro de pequeños subconjuntos. Este seccionamiento puede ser arbitrario y no es necesario obtener el conjunto de grupos óptimo del conjunto de datos de entrada, sino únicamente seccionar el espacio de datos de entrada dentro de un número relativamente grande de subconjuntos que cubran el espacio de datos de entrada en su totalidad. De esta forma, en esta primera fase del algoritmo, la carga computacional al procesar los datos está dada por el número de grupos o secciones que cubren el conjunto de datos

y no por el conjunto de datos entero, provocando que el tiempo de entrenamiento de la SVM sea mucho menor. A partir de la primera etapa de clasificación, obtenemos un conjunto de vectores soporte que son centros de grupos, cuyos elementos dentro de estos grupos tienen una mayor probabilidad de ser vectores soporte, i.e. contienen los elementos más representativos del conjunto de datos. Una vez obtenidos estos vectores soporte que representan los centros de los grupos o secciones más representativas del conjunto de datos entero empleamos un proceso de desagrupamiento sobre estos grupos y obtenemos un subconjunto de datos reducido con los datos más importantes para una SVM, reduciendo enormemente el tiempo de entrenamiento de las SVM.

A lo largo de la tesis se muestra que la velocidad de entrenamiento y generalización de las SVM puede ser incrementada eliminando subconjuntos de datos no representativos a partir del conjunto de datos original y centrar la mayor carga de trabajo sobre aquellos subconjuntos más representativos del conjunto de datos entero, lo cual es crucial en conjuntos de datos grandes. Esto es posible de realizar implementando métodos de agrupamiento y seccionamiento de los conjuntos de datos de entrada.

1.1. Estado del arte

Debido a sus buenos fundamentos teóricos y su buena capacidad de generalización las SVM han llegado a ser en los últimos años uno de los métodos de clasificación más utilizado. Sin embargo, las SVM requieren grandes cantidades de tiempo computacional en la fase de entrenamiento cuando el número de datos de entrenamiento es muy grande. Un gran número de implementaciones basadas en SVM han sido desarrolladas con el objetivo de afrontar este problema. Muchos trabajos de investigación han tratado de encontrar posibles métodos para implementar SVM en grandes conjuntos de datos. Generalmente estos métodos pueden ser divididos en dos tipos:

1. el primer tipo consiste en encontrar candidatos a vectores soporte, reduciendo el conjunto de trabajo y empleando técnicas de búsqueda, de tal forma que la

SVM modificada sea capaz de entrenar conjuntos de datos con únicamente datos representativos reduciendo el tiempo de entrenamiento a un tiempo aceptable;

2. el segundo tipo consiste en descomponer el conjunto de datos de entrada en conjuntos pequeños, de tal forma que las SVM clásicas puedan ser utilizadas.

El algoritmo de clasificación más empleado para grandes conjuntos de datos es el algoritmo de Optimización Mínima Secuencial (SMO) [67], el cual es obtenido a partir de la idea de descomponer un conjunto de datos a su extremo y optimizar un subconjunto mínimo de dos puntos únicamente en cada iteración. El poder de esta técnica reside en el hecho de que el problema de optimización para dos puntos admite una solución analítica, eliminando la necesidad de usar un optimizador de programación cuadrática iterativo como parte del algoritmo [13][36][81]. Pavlov aplica el *boosting* al algoritmo de SMO de Platt con el objetivo de escalar y agilizar el entrenamiento de las SVM [66]. La principal idea de *boosting* es entrenar una secuencia de clasificadores de tal forma que cada clasificador posterior concentre la mayoría de los errores realizados por los anteriores. Esto es llevado a cabo asignando una etiqueta de probabilidad a cada patrón de entrenamiento y manteniéndola durante toda la fase de entrenamiento. Las reglas para actualizar estas etiquetas de probabilidad están totalmente especificadas por el algoritmo *boosting*. El peor desempeño del clasificador previo sobre un conjunto de ejemplos en particular será la más alta probabilidad que éste obtendrá. Los resultados reportados en [43] demuestran que una ventaja computacional común puede ser obtenida usando una estrategia recursiva para grandes conjuntos de datos, tales como aquellas involucradas en aplicaciones de minería de datos y categorización de datos. La cuantización vectorial es utilizada en [46] para reducir un gran conjunto de datos reemplazando ejemplos por prototipos. El tiempo de entrenamiento para elegir parámetros óptimos es reducido gratamente.

Una heurística muy empleada es "*Chunking*". Ésta emplea un subconjunto arbitrario de datos llamado "*chunk*" y entrena una SVM sobre esta porción de datos. El algoritmo retiene los vectores soporte del *chunk* mientras que descarta todos los demás puntos

del *chunk*, una vez realizado esto, emplea los vectores soporte encontrados para probar en los demás puntos que no pertenecen al *chunk* inicial. En cada iteración, el algoritmo selecciona un conjunto de k puntos que viola las condiciones de Karush-Kuhn-Tucker [40] [44]. Estos k puntos son adheridos a los vectores soporte del *chunk* anterior para formar un nuevo *chunk*. Este procedimiento es iterado hasta que se satisface un criterio de paro, aunque ninguna prueba teórica de la convergencia de este algoritmo ha sido realizada, en la práctica trabaja muy bien y hace posible entrenar conjuntos de datos de tamaño mediano. En [52] propone un enfoque basado en una técnica de aprendizaje incremental y múltiples aproximaciones empleando SVM. Algunas otras técnicas incluyen Selección Aleatoria [5] [79] [88] y Rocchio Bundling [81]; éstas muestrean un pequeño número de datos de entrenamiento a partir del conjunto de datos original de tal forma que sea maximizado el grado de aprendizaje. Sin embargo, los algoritmos de muestreo podrían simplificar el conjunto de datos de entrenamiento perdiendo los beneficios de usar SVM, especialmente cuando la distribución de probabilidad de datos de entrenamiento y prueba son diferentes. Un enfoque geométrico fue propuesto por Mavroforakis et al [54], el algoritmo propuesto reduce el tamaño de los patrones de entrenamiento al encontrar pares de puntos cercanos empleando "convex hulls". En [22], los autores introducen un paso de optimización paralela donde un conjunto de matrices son empleadas para aproximar la matriz original de tal forma que la clasificación con SVM puede ser dividida en cientos de pequeños problemas de optimización.

Los métodos de agrupamiento son una efectiva herramienta para reducir el tamaño de los conjuntos de datos. El uso de técnicas de agrupamiento antes de utilizar el clasificador es una estrategia interesante para problemas con grandes conjuntos de datos. Aunque muchos métodos para resolver el problema de optimización de las SVM mediante métodos de agrupamiento están disponibles en la literatura, se listan aquí solo algunas técnicas interesantes usadas para entrenar SVM con grandes conjuntos de datos, tales como CB-SVM [99], CB-SOCP [77], CT-SVM [42], y RS-MCS [11].

CB-SVM [99] aplica micro-agrupamiento jerárquico que escanea el conjunto de datos entero una sola vez. El método propuesto escala bien para grandes conjuntos de datos y

las precisiones obtenidas mediante este método son comparables a las otras implementaciones de SVM. Sin embargo, el micro-agrupamiento empleado en CB-SVM es demasiado dependiente de la dimensión del conjunto de datos de entrada y puede no desempeñarse bien en conjuntos de datos altamente dimensionales, además el algoritmo está diseñado para trabajar únicamente con kernel lineal.

CB-SOCP [77] es diseñado para grandes conjuntos de datos. Sus autores asumen que las densidades de clase de los datos de entrada pueden ser modelados usando una combinación de modelos. El algoritmo emplea BIRCH [100] con el objetivo de estimar las estadísticas de los componentes. El método propuesto es escalable para grandes conjuntos de datos y las precisiones obtenidas sobre diferentes conjuntos de datos empleando CB-SOCP son comparables a las obtenidas con otras implementaciones de SVM. Sin embargo, el algoritmo CB-SOCP muestrea aleatoriamente el conjunto de datos de entrada, este proceso podría afectar el proceso de entrenamiento de las SVM, especialmente cuando la probabilidad de distribución de los datos de entrenamiento y prueba son diferentes.

CT-SVM[42] aplica técnicas de reducción mediante análisis de agrupamiento para encontrar vectores soporte relevantes con el objetivo de agilizar el proceso de entrenamiento. El algoritmo construye un árbol de agrupamiento jerárquico para cada clase a partir del conjunto de datos de entrada; esto lo hace de forma iterativa en varias etapas de entrenamiento. En cada etapa, una SVM es entrenada sobre los nodos de cada árbol. Los vectores soporte del clasificador son empleados como conocimiento previo que determinan el crecimiento del árbol. Únicamente a los vectores soporte les es permitido crecer en el árbol, mientras los puntos que no son vectores soporte son eliminados. Este método es escalable para grandes conjuntos de datos, sin embargo el algoritmo es sensible en conjuntos de datos con ruido y susceptible a conjuntos de datos incompletos.

RS-MCS [11] emplea pares de puntos "espejos" y un sistema de clasificador múltiple para reducir el tiempo de entrenamiento de una SVM. Los autores desarrollan un enfoque mediante agrupamiento *K-Means* con el objetivo de seleccionar y combinar un número dado de clasificadores, los cuales toman en cuenta precisión y eficiencia. Las precisiones

Tabla 1.1: Comparación entre algunos algoritmos del estado del arte

Algoritmo	TE	TK	TCD
SMO	Conjunto de trabajo	lineal y no lineal	Mediano
Boosting	Conjuntos de trabajo pequeño y SMO	lineal	Mediano
Chunking	Conjuntos de trabajo pequeño	lineal y no lineal	Mediano
CB-SVM	Agrupamiento jerárquico	lineal	Grande
CB-SOCP	Agrupamiento con BIRCH	lineal y no lineal	Grande
CT-SVM	Agrupamiento jerárquico	lineal y no lineal	Grande
RS-MCS	Agrupamiento K-means	lineal y no lineal	Mediano

TE = Técnica empleada para reducir el tiempo de entrenamiento, TK = Tipo de kernel, TCD = Tamaño del conjunto de datos.

obtenidas empleando este algoritmo son buenas, sin embargo este método trabaja con conjuntos de datos de pequeño y mediano tamaño.

1.2. Motivación

La clasificación de grandes conjuntos de datos ha tomado gran importancia en los últimos años debido al intenso progreso de la computación a otras tecnologías relacionadas, que han provocado que enormes cantidades de datos con múltiples características sean continuamente almacenadas en bases de datos.

La mayoría de los métodos de agilización de SVM, reportados en la literatura, procesan el conjunto de datos en su totalidad con el propósito de obtener una buena precisión de clasificación al emplear los resultados obtenidos sobre conjuntos de prueba. Sin embargo, las SVM poseen una gran desventaja como resultado de la excesiva carga

computacional presentada al procesar todos los datos.

Muchos algoritmos de clasificación con SVM han sido propuestos en la literatura con el objetivo de enfrentar estas desventajas, sin embargo el tiempo de entrenamiento aún sigue siendo muy alto y en algunos casos, en los que se logra reducir significativamente el tiempo de entrenamiento, la precisión lograda no es muy deseable. Es por ello que el diseño de algoritmos que resuelvan este problema es un reto muy necesario. En este trabajo de tesis, se proponen algunos algoritmos que tratan de resolver este problema de intercambio recíproco entre tiempo de entrenamiento y precisión.

1.3. Objetivos

El objetivo principal de esta tesis es desarrollar algoritmos de clasificación empleando SVM, que sean una alternativa viable en la literatura para resolver problemas de clasificación en donde el conjunto de datos de entrada es muy grande. Los algoritmos propuestos deben ser capaces de resolver problemas de clasificación cuyo tiempo de entrenamiento y precisión sea competitivo con respecto a los algoritmos representativos del estado del arte en el área.

Los objetivos específicos son los siguientes:

1. Desarrollar algoritmos de clasificación binaria para grandes conjuntos de datos empleando SVM, reduciendo el tiempo de entrenamiento que conlleva emplear SVM en conjuntos de datos grandes

2. Desarrollar algoritmos de clasificación para grandes conjuntos de datos con múltiples clases

3. Obtener la complejidad algorítmica de los algoritmos propuestos

4. Implementar un algoritmo para clasificación de intrones y exones en secuencias de ADN.

1.4. Contribuciones

Las principales contribuciones derivadas de este trabajo de investigación son las siguientes:

1. Desarrollo de estrategias de SVM para clasificar conjuntos de datos grandes, las estrategias son; SVM en dos etapas basadas en agrupamiento difuso ($FCM-SVM^2$), SVM en dos etapas basadas en esferas de cerradura mínima ($MEB-SVM^2$) y SVM en dos etapas basadas en selección aleatoria ($RS-SVM^2$). Las estrategias propuestas descomponen el conjunto de datos de entrada mediante algoritmos de agrupamiento o selección aleatoria, los resultados obtenidos en cuanto a tiempo de entrenamiento y precisión de clasificación son competitivos con los algoritmos representativos del estado del arte en el área.

2. Análisis de complejidad de algoritmos desarrollados.

3. Desarrollo de un multiclasificador para grandes conjuntos de datos.

4. Un algoritmo para clasificación de intrones y exones en secuencias de ADN. El algoritmo propuesto es competitivo en comparación con los algoritmos empleados en el área.

1.5. Organización de la tesis

El presente documento está organizado de la siguiente manera: en el Capítulo 2 iniciamos con un análisis del problema de clasificación general, se presentan los conceptos básicos de las SVM, se describen formalmente las SVM para el caso linealmente separable y para el caso linealmente no separable. En el Capítulo 3, se presenta la problemática asociada a las SVM cuando se trabaja sobre grandes conjuntos de datos y se proponen algoritmos de clasificación en dos etapas basados en SVM. El Capítulo 4, se muestran los resultados experimentales de los algoritmos propuestos. En el Capítulo 5, se muestra

el análisis de desempeños de los algoritmos propuestos. En el Capitulo 6 se propone un algoritmo para el caso de múltiples clases. En el Capítulo 7, se propone un algoritmo basado en árboles de decisión. El algoritmo es empleado para clasificar exones dentro de secuencias de ADN en células eucariotas. Finalmente, el Capítulo 8 presenta las conclusiones y trabajo futuro.

Capítulo 2

Máquinas de Vectores Soporte (SVM)

La clasificación de patrones se define como la tarea de categorizar algún objeto dentro de una de las categorías dadas llamadas clases, a partir de un conjunto de patrones asociados a cada objeto. Usamos el término "patrón" para denotar un vector de datos x de dimensión p, donde $x = (x_1, \ldots, x_p)^T$ cuyos componentes x_i son las medidas de las características de un objeto. Estas características son las variables especificadas por el investigador, debido a que por lo regular tienen un peso importante en los resultados de la clasificación.

En general, existen dos enfoques principales de clasificación: *clasificación supervisada* y *clasificación no supervisada.* La clasificación no supervisada también es referida frecuentemente como agrupamiento. En este tipo de clasificación, los datos no son etiquetados y se desean encontrar grupos en los datos que se distingan unos de otros a partir de las características. En la clasificación supervisada tenemos un conjunto de datos de prueba, cada uno de estos consiste de medidas sobre un conjunto de variables y asociado a cada dato una *etiqueta* que define la clase del objeto.

Las redes neuronales, árboles de decisión y SVM son clasificadores de aprendizaje supervisado. Los métodos de aprendizaje supervisado emplean un conjunto de pares

entrada-salida, estos clasificadores adquieren una función de decisión que asocia a un nuevo dato una etiqueta de clase dentro de las clases dadas.

En este Capítulo son definidas las características teóricas de las SVM para problemas de clasificación con dos clases. Primero, definimos las funciones decisión y su importancia al generalizar, después definimos las Máquinas de Vectores Soporte con margen duro, para conjuntos de datos de entrenamiento linealmente separables en el espacio de entrada. Una vez concluido esto, nos extendemos a el caso linealmente no separable y es necesario trasladar el espacio de datos de entrada a un espacio de características altamente dimensional con el propósito de separar linealmente el espacio de características.

2.1. Funciones de decisión

Consideremos el problema de clasificación de un punto cuyas características están dadas por el vector x tal que $x = (x_1, \ldots, x_p)^T$ y este pertenece a una de dos clases posibles. Supongamos que tenemos las funciones $f_1(x)$ y $f_2(x)$ que definen las clases 1 y 2 y nosotros clasificamos al punto x dentro de la clase 1 si

$$f_1(x) > 0,\ f_2(x) < 0,$$

o clasificamos al punto x dentro de la clase 2 si

$$f_1(x) < 0,\ f_2(x) > 0,$$

A estas funciones las llamamos funciones de decisión. Al proceso de encontrar las funciones de decisión a partir de pares de entrada-salida es llamado *entrenamiento.* Los métodos convencionales de entrenamiento determinan las funciones de decisión de tal forma que cada par entrada-salida sea correctamente clasificado dentro de la clase a la que pertenece. La Figura 2.1 muestra un ejemplo. Asumiendo que los cuadros pertenecen a la clase 1 y los círculos pertenecen a la clase 2, resulta claro que los datos de entrenamiento no se intersectan en ningún momento y es posible trazar una línea separando los datos

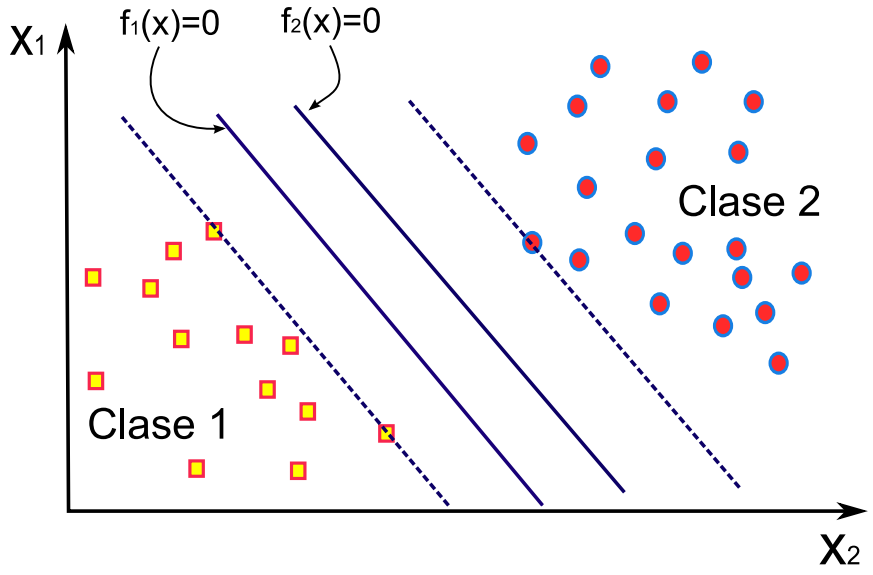

Figura 2.1: Funciones de decisión

de manera perfecta. Sin embargo, ya sea que la función de decisión $f_1(x)$ o la función $f_2(x)$ se muevan hacia la línea punteada de su propio lado, el conjunto de datos de entrenamiento aún sigue siendo correctamente clasificado, dándonos la certeza de que es posible encontrar un conjunto infinito de hiperplanos que correctamente clasifiquen los datos de entrenamiento. Sin embargo, es claro que la precisión de clasificación al generalizar será directamente afectada por la posición de las funciones de decisión.

Las SVM a diferencia de otros métodos de clasificación consideran esta desventaja y encuentra la función de decisión de tal forma que la distancia entre los datos de entrenamiento es maximizada. Esta función de decisión es llamada *función de decisión óptima* o *hiperplano de decisión óptima [15].*

2.2. Funciones de decisión en las SVM

Recientemente ha habido un incremento impresionante en el número de artículos de investigación sobre SVM. Las SVM han sido aplicadas exitosamente a un gran número de aplicaciones yendo desde identificación de partículas, identificación de rostros y categorización de texto hasta bioinformática y medicina. El enfoque es motivado por la teoría de aprendizaje estadístico [91] [92]. Las SVM producen modelos matemáticos elegantes que son geométricamente intuitivos y teóricamente bien fundamentados.

La principal motivación de las SVM es separar varias clases en el conjunto de entrenamiento con una superficie que maximice el margen entre estas. Esta es una implementación del principio de minimización estructural que permite minimizar una cota sobre el error de generalización de un modelo, en lugar de minimizar el error medio cuadrático sobre el conjunto de datos de entrenamiento, que es la filosofía que usan a menudo los métodos de minimización de riesgo empírico.

Entrenar una SVM requiere un conjunto de n ejemplos. Cada ejemplo consiste de un vector de entrada x_i y una etiqueta y_i asociada al vector de entrada. La función de la SVM que tiene que ser entrenada con los ejemplos contiene n parámetros libres, los llamados multiplicadores de Lagrange positivos $\alpha_i, i = 1, \ldots, n$. Cada α_i es una medida de cuanto, el correspondiente ejemplo de entrenamiento influye en la función. La mayoría de los ejemplos no afectan la función y consecuentemente la mayoría de los α_i son cero. De manera general, Las SVM y los conceptos fundamentales que las describen son planteados formalmente en esta sección:

2.2.1. Caso linealmente separable

Consideremos el problema de clasificación binaria en donde los datos de entrenamiento son dados como

$$(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l), x \in \mathbb{R}^n, y \in \{+1, -1\} \tag{2.1}$$

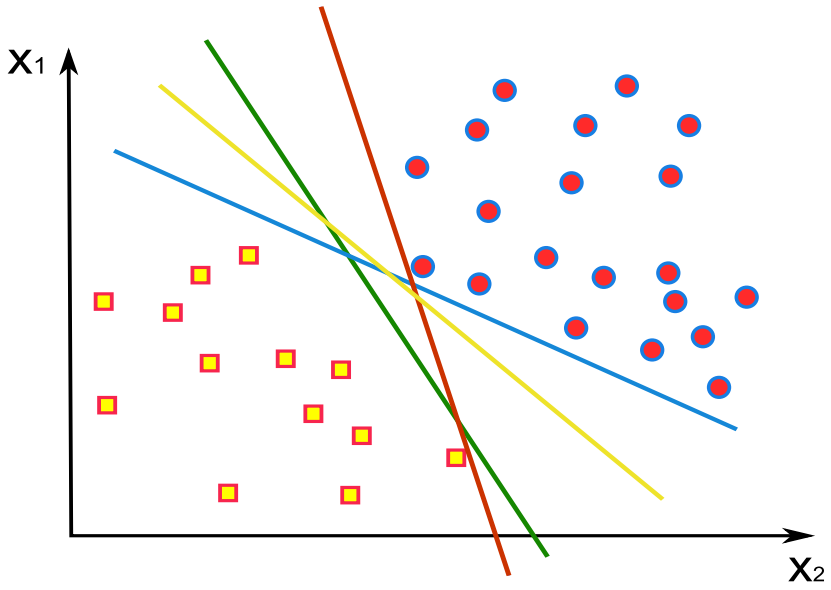

Figura 2.2: Clasificador convencional

por razones de visualización, consideramos el caso de un espacio de entrada bidimensional, i.e., $x \in \mathbb{R}^2$. Los datos son linealmente separables y existen diferentes hiperplanos que pueden realizar la separación. La Figura 2.2 muestra varios hiperplanos de decisión que separan perfectamente el conjunto de datos de entrada. Es claro que existe un número infinito de hiperplanos que podrían realizar este trabajo. Sin embargo, la habilidad de generalización depende de la localización del hiperplano de separación y el hiperplano con máximo margen es llamado *hiperplano de separación óptima [15]*. La cota de decisión, i.e. la línea que separa el espacio de entrada es definida por la ecuación $w^T x_i + b = 0$. Sin embargo, el problema radica en encontrar la mejor cota de decisión, i.e., la función de separación óptima.

El caso más simple de SVM es el caso linealmente separable en el espacio de características. Optimizamos el margen geométrico fijando para ello el margen funcional $\kappa_i = 1$ (también llamado *Hiperplano Canónico [20]*), por lo tanto, el clasificador lineal

$y_i = \pm 1$,

$$\begin{aligned} \langle w \cdot x^+ \rangle + b &= 1 \\ \langle w \cdot x^- \rangle + b &= -1 \end{aligned} \tag{2.2}$$

Estos pueden ser combinados dentro de un conjunto de desigualdades:

$$y_i(\langle w \cdot x_i \rangle + b) \geq 1 \ \forall i \tag{2.3}$$

el margen geométrico de x^+ y x^- es

$$\begin{aligned} \gamma_i &= \tfrac{1}{2}\left(\left\langle \tfrac{w}{\|w\|} \cdot x^+ \right\rangle - \left\langle \tfrac{w}{\|w\|} \cdot x^- \right\rangle\right) \\ &= \frac{1}{2\|w\|}\left[\langle w \cdot x^+ \rangle - \langle w \cdot x^- \rangle\right] \\ &= \frac{1}{\|w\|} \end{aligned} \tag{2.4}$$

donde w define el hiperplano de separación óptima y b es el sesgo. La distancia entre el hiperplano de separación y el dato de entrenamiento más cercano al hiperplano, es llamada margen. La habilidad de generalización depende de la localización del hiperplano de separación y el hiperplano con máximo margen es llamado hiperplano de separación óptima. Es intuitivamente claro que la habilidad de generalización es maximizada si el hiperplano de separación óptima es seleccionado como el hiperplano de separación. Optimizar el margen geométrico significa minimizar la norma del vector de pesos. Al resolver el problema de programación cuadrática tratamos de encontrar el hiperplano óptimo y dos hiperplanos (H_1 y H_2) paralelos. Las distancias entre H_1 y H_2 es maximizada y no existe ningún dato entre los dos hiperplanos. Cuando la distancia entre H_1 y H_2 es maximizada, algunos puntos de datos pueden estar sobre H_1 y algunos puntos de datos pueden estar sobre H_2. Estos puntos de datos son llamados *vectores soporte [15] [20]*, ya que participan de forma directa en definir el hiperplano de separación, los otros puntos pueden ser removidos o cambiados sin cruzar los planos H_1 y H_2 y no modificarán de alguna forma la habilidad de generalización del clasificador, i.e., la solución de una

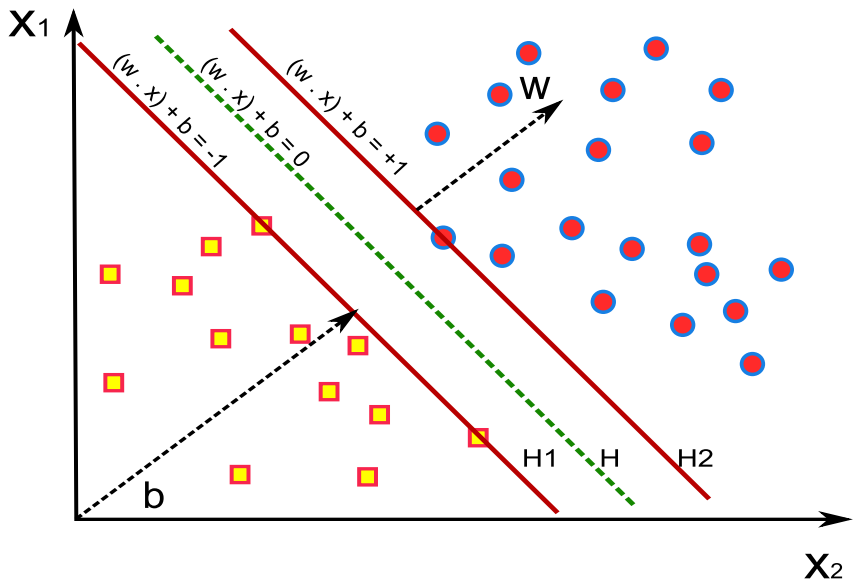

Figura 2.3: Clasificador óptimo

SVM está dada únicamente por éste pequeño conjunto de vectores soporte. Cualquier hiperplano puede ser representado mediante w, x y b, donde w es un vector perpendicular al hiperplano. La Figura 2.3 muestra la representación geométrica del problema de programación cuadrática mostrando H (separador óptimo) y los hiperplanos H_1 y H_2. De esta forma, el problema original de optimización queda de la siguiente manera.

Proposición 2.1 *Para el caso linealmente separable* $S = [(x_1, y_1) \cdots (x_l, y_l)]$ *, si el hiperplano* (w, b) *es la solución de*

$$\begin{aligned} &\min_{w,b} \langle w \cdot w \rangle = \|w\|^2 \\ &\textit{sujeto a: } y_i \left(\langle w \cdot x_i \rangle + b \right) \geq 1 \end{aligned} \tag{2.5}$$

entonces el hiperplano tiene un margen máximo (geométrico) $\gamma = \frac{1}{\|w\|}$.

Ahora cambiamos al problema dual utilizando la formulación de Lagrange. Existen dos razones para hacer esto. La primera radica en el hecho de que las condiciones dadas

en la ecuación (2.3) serán reemplazadas por multiplicadores de Lagrange, que son mucho más fáciles de manejar. La segunda proviene de que, en la reformulación del problema, los datos de entrenamiento únicamente aparecerán en la forma de producto punto entre vectores. Esta es una propiedad fundamental que permitirá generalizar el procedimiento en el caso no lineal. De esta manera, el Lagrangiano está dado por:

$$L(w,b,\alpha) \equiv \frac{1}{2}\langle w \cdot w\rangle - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\langle w \cdot x_i\rangle + b\right) - 1\right]$$

donde α_i son multiplicadores de Lagrange.

El dual es encontrado en dos pasos: primero, diferenciando con respecto a w y b

$$\frac{\partial L(w,b,\alpha)}{\partial w} = w - \sum\nolimits_{i=1}^{l} \alpha_i y_i x_i = 0 \rightarrow w = \sum\nolimits_{i=1}^{l} \alpha_i y_i x_i$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = -\sum\nolimits_{i=1}^{l} \alpha_i y_i = 0 \rightarrow \sum\nolimits_{i=1}^{l} \alpha_i y_i = 0$$

y segundo, resustituyendo las relaciones obtenidas en el Lagrangiano original

$$L\left(w,b,\alpha\right) = \tfrac{1}{2}\langle w \cdot w\rangle - \textstyle\sum_{i=1}^{l} \alpha_i \left[y_i \left(\langle w \cdot x_i\rangle + b\right) - 1\right]$$

$$= \frac{1}{2}\left\langle \sum\nolimits_{i=1}^{l} \alpha_i y_i x_i \cdot \sum\nolimits_{i=1}^{l} \alpha_i y_i x_i \right\rangle - \sum\nolimits_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \left(\langle x_j \cdot x_i\rangle + b\right) - \sum\nolimits_{i=1}^{l} \alpha_i$$

$$= \frac{1}{2}\sum\nolimits_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle x_i \cdot x_j\rangle - \sum\nolimits_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle x_j \cdot x_i\rangle - \sum\nolimits_{i=1}^{l} \alpha_i y_i b + \sum\nolimits_{i=1}^{l} \alpha_i$$

$$= -\frac{1}{2}\sum\nolimits_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle x_i \cdot x_j\rangle + \sum\nolimits_{i=1}^{l} \alpha_i$$

Aquellos puntos para los cuales $\alpha_i > 0$ son llamados "vectores soporte" y quedan en uno de los hiperplanos H_1, H_2. En todos los otros puntos de entrenamiento $\alpha_i = 0$ y soportan o quedan sobre H_1 o H_2 de tal forma que las condiciones de la ecuación (2.3) se cumplen. Los vectores soporte son los elementos críticos del conjunto de entrenamiento y estos son los más cercanos a la cota de decisión.

Comentario 2.1 *Al entrenar el conjunto inicial de datos obtenemos un hiperplano, que separa perfectamente estos datos y es definido por un pequeño conjunto de vectores soporte. Si todos los demás puntos fueran eliminados (o desplazados alrededor sin cruzar H_1 o H_2) y el entrenamiento fuera repetido, se encontraría el mismo hiperplano de separación definido por el mismo conjunto de vectores soporte.*

Por lo tanto, el problema original de optimización queda de la siguiente manera.

Proposición 2.2 *Para el caso linealmente separable $S = [(x_1, y_1) \cdots (x_l, y_l)]$, si α_i^* es la solución del problema de optimización cuadrático*

$$\begin{gathered} \max_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle x_i \cdot x_j \rangle + \sum_{i=1}^{l} \alpha_i \\ \text{sujeto a: } \sum_{i=1}^{l} \alpha_i y_i = 0 \end{gathered}$$

entonces $\|w\|^2$ comprende el mínimo $w^* = \sum_{i=1}^{l} \alpha_i^* y_i x_i$ y el margen geométrico $\gamma^* = \frac{1}{\|w^*\|}$ es maximizado.

2.2.2. Condiciones de Karush-Kuhn-Tucker

Las condiciones de Karush-Kuhn-Tucker (KKT) [42] [44] juegan un rol muy importante en la teoría de optimización, ya que dan las condiciones para obtener una solución óptima a un problema de optimización general.

Teorema 2.1 *Dado un problema de optimización con dominio convexo $\Omega \subseteq \mathbb{R}^n$,*

$$\begin{array}{ll} \textit{minimizar} & f(w), \qquad w \in \Omega \\ \textit{sujeto a} & g_i(w) \leq 0, i = 1, \ldots, k, \\ & h_i(w) = 0, i = 1, \ldots, m, \end{array}$$

con $f \in C^1$ convexa, las condiciones necesarias y suficientes para que un punto

normal w^* *sea un óptimo son la existencia de* α^*, β^* *tal que*

$$\begin{aligned}
&\frac{\partial L(w^*, \alpha^*, \beta^*)}{\partial w} = 0 \\
&\frac{\partial L(w^*, \alpha^*, \beta^*)}{\partial \beta} = 0 \\
&\alpha_i^* g_i(w^*) = 0, i = 1, \ldots, k, \\
&g_i(w^*) \leq 0, i = 1, \ldots, k, \\
&\alpha_i^* \geq 0, i = 1, \ldots, k.
\end{aligned}$$

De las condiciones de KKT tenemos que si el conjunto de entrenamiento es linealmente separable, se verifica que

$$\|w^*\|^2 = \langle w^* \cdot w^* \rangle = \left(\sum_{i \in sv} \alpha_i^* \right)^{-\frac{1}{2}}$$

Por lo tanto, la distancia máxima de un hiperplano es:

$$\frac{1}{\|w^*\|} = \left(\sum_{i \in sv} \alpha_i^* \right)^{-\frac{1}{2}}$$

2.2.3. Hiperplanos con márgenes blandos

El problema de aprendizaje presentado anteriormente es válido para el caso donde los datos son linealmente separables, que significa que el conjunto de datos de entrenamiento no tiene intersecciones. Sin embargo, este tipo de problemas son raros en la práctica. Al mismo tiempo, existen algunos ejemplos en los que el hiperplano de separación lineal puede dar buenos resultados aún cuando los datos se intersectan. Sin embargo, las soluciones de programación cuadrática como están dadas anteriormente no pueden ser usadas en el caso de intersección ya que la condición $y_i(\langle w \cdot x_i \rangle + b) \geq 1 \ \forall i$ no puede ser satisfecha en el caso de intersección (ver Figura 2.4). Los puntos que se encuentran en la intersección no pueden ser correctamente clasificados y para cualquier dato mal clasificado x_i, su correspondiente α_i tenderá a infinito.

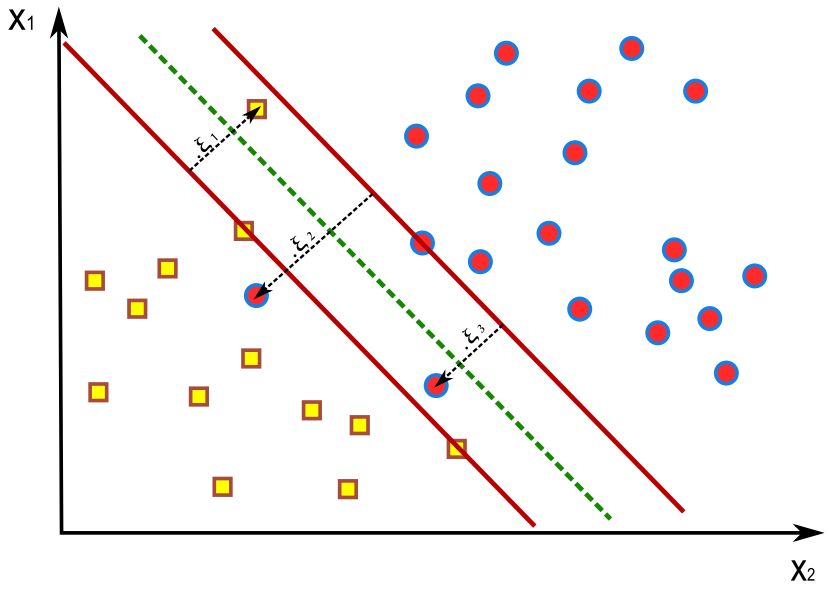

Figura 2.4: Hiperplanos con margen blando

Para encontrar un clasificador con margen máximo, el algoritmo presentado anteriormente deberá ser cambiado permitiendo un margen blando, por lo tanto es necesario introducir variables *flojas* no negativas $\xi_i (\geq 0)$ en la ecuación (2.3)

$$y_i(\langle w^T \cdot x_i \rangle + b) \geq 1 - \xi_i \qquad \forall i \tag{2.6}$$

Mediante las variables ξ_i, la solución factible siempre existe. Para los datos de entrenamiento x_i, si $0 < \xi_i < 1$, los datos no poseen el margen máximo, pero pueden ser correctamente clasificados. Por otro lado, el ancho de este margen blando puede ser controlado por el parámetro de penalización C, que determina la relación entre el error de entrenamiento y la dimensión Vapnik-Chervonenkis del módulo.

Definición 2.1 *(Dimensión Vapnik-Chervonenkis -VC-) La dimensión VC describe la capacidad de un conjunto de funciones implementadas en una máquina de aprendizaje. Para clasificación binaria h es el máximo número de puntos en el que pueden ser sep-*

aradas dos clases en todas la 2^h formas posibles usando las funciones de la máquina de aprendizaje.

Un C grande proporciona un pequeño número de errores de clasificación y un gran $w^T w$. Es claro que tomando $C = \infty$ requiere que el número de datos mal clasificados sea cero. Sin embargo, en este caso no es posible, ya que el problema puede ser factible únicamente para algún valor $C < \infty$. Introduciendo "variables flojas" no negativas $\xi_i (i = 1, l)$ al problema de optimización, ahora en lugar de la condiciones de la ecuación (2.5) el hiperplano de separación deberá satisfacer

$$\begin{array}{c} \min\limits_{w,b,\xi_i} \langle w \cdot w \rangle + C \sum_{i=1}^{l} \xi_i^2 \\ \text{sujeto a: } y_i \left(\langle w \cdot x_i \rangle + b \right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{array} \tag{2.7}$$

i.e., sujeto a

$$\begin{array}{rcl} \langle w \cdot x_i \rangle + b & \geq & +1 - \xi_i, \text{ para } y_i = +1, \xi_i \geq 0 \\ \langle w \cdot x_i \rangle + b & \leq & -1 + \xi_i, \text{ para } y_i = -1, \xi_i \geq 0 \end{array} \tag{2.8}$$

Si $\xi_i < 0$, $y_i \left(\langle w \cdot x_i \rangle + b \right) \geq 1 - \xi_i \geq 1$, por lo tanto, no consideramos la condición $\xi_i < 0$.

Para el máximo margen blando con Norma-2 (con la diagonal $\frac{1}{C}\delta_{ij}$) el Lagrangiano original está dado por:

$$L\left(w, b, \xi_i, \alpha\right) = \frac{1}{2} \langle w \cdot w \rangle - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\langle w \cdot x_i \rangle + b \right) - 1 + \xi_i \right] + \frac{C}{2} \sum_{i=1}^{l} \xi_i^2$$

El dual es encontrado en dos pasos: de la misma manera que en el caso linealmente separable primero diferenciando con respecto a w y b, y después resustituyendo en el Lagrangiano original, de tal forma que el problema dual sería

$$\begin{array}{c} \max\limits_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \left[\langle x_i \cdot x_j \rangle + \frac{1}{C} \delta_{ij} \right] + \sum_{i=1}^{l} \alpha_i \\ \text{sujeto a: } \sum_{i=1}^{l} \alpha_i y_i = 0 \end{array}$$

La condición de Kuhn-Tucker es

$$\alpha_i^* \left[y_i \left(\langle w^* \cdot x_i \rangle + b^* \right) - 1 + \xi_i \right] = 0$$

Esto es, el problema de optimización cuadrática es prácticamente el mismo que en el caso separable con la única diferencia de las cotas modificadas de los multiplicadores de Lagrange α_i. El parámetro C es determinado por el usuario. La selección de una apropiada C es realizada experimentalmente usando alguna técnica de validación cruzada[20] [15] [32].

2.2.4. Kernels

En una SVM, el hiperplano óptimo es determinado para maximizar su habilidad de generalización. Pero, si lo datos de entrenamiento no son linealmente separables, el clasificador obtenido puede no tener una alta habilidad de generalización, aún cuando los hiperplanos sean determinados óptimamente i.e., para maximizar el espacio entre clases, el espacio de entrada original es transformado dentro de un espacio altamente dimensional llamado "espacio de características".

La idea básica en diseño de SVM no lineales es transformar los vectores de entrada $x \in \mathbb{R}^n$ dentro de vectores $\Phi(x)$ de un espacio de características altamente dimensional [20] F (donde Φ representa el mapeo: $\mathbb{R}^n \rightarrow \mathbb{R}^f$) y resolver el problema de clasificación lineal en este espacio de características

$$x \in \mathbb{R}^n \rightarrow \Phi(x) = \left[\phi_1(x), \phi_2(x), \ldots, \phi_n(x) \right]^T \in \mathbb{R}^f$$

El conjunto de hipótesis que consideraremos serán funciones de tipo

$$f(x) = \sum_{i=1}^{l} w_i \phi_i(x) + b \tag{2.9}$$

donde $\phi : X \rightarrow F$ es un mapeo no lineal desde un espacio de entrada a un *espacio de características*, i.e., el procedimiento de aprendizaje consiste de dos pasos: primero, un

mapeo no lineal transforma los datos dentro de un espacio de características F y después, una máquina lineal es utilizada para clasificar los datos en un espacio de características.

Como se vio anteriormente, una propiedad de las máquinas de aprendizaje lineal es que éstas pueden ser expresadas en una representación dual, esto significa que la ecuación (2.9) puede ser expresada como una combinación lineal de los puntos de entrenamiento. Por lo tanto, la regla de decisión puede ser evaluada usando productos punto

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i \langle \phi(x_i) \cdot \phi(x) \rangle + b$$

Si se tiene una forma de capturar el producto $\langle \phi(x_i) \cdot \phi(x) \rangle$ en el espacio de características, directamente como una función de los puntos de entrada originales, esto hace posible unir los dos pasos necesarios para construir una máquina de aprendizaje no-lineal. A este método de cómputo directo se le llama función kernel [92].

Definición 2.2 *Un kernel es una función K, tal que para todo $x, z \in X$*

$$K(x, z) = \langle \phi(x) \cdot \phi(z) \rangle$$

donde ϕ es un mapeo de X a un espacio de características F.

La clave del enfoque es encontrar una función kernel que pueda ser evaluada eficientemente. Una vez que tenemos tal función de decisión, la regla puede ser evaluada

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i K \langle x_i \cdot x \rangle + b$$

2.2.5. Condición de Mercer

El teorema de Mercer [21] [91] provee una caracterización de cuando una función $K(\mathbf{x}, \mathbf{z})$ es un kernel. Dado un espacio de entrada finito $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ y suponiendo que $K(\mathbf{x}, \mathbf{z})$ es una función simétrica de X, entonces la matriz

$$\mathbf{K} = \left(K \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)\right)_{i,j=1}^{n}$$

Ya que $\mathbf{K}$ es simétrica existe una matriz ortogonal $\mathbf{V}$ tal que $\mathbf{K} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}'$, donde $\mathbf{\Lambda}$ es la matriz diagonal que contiene los eigenvalores λ_t de $\mathbf{K}$, con sus correspondientes eigenvectores $\mathbf{v}_t = (v_{ti})_{i=1}$. Asumiendo que todos los eigenvalores son no-negativos y considerando el mapeo

$$\phi : \mathbf{x}_i \mapsto (\sqrt{\lambda_t} v_{ti})_{t=1}^n \in \mathbb{R}^n, i = 1, \ldots, n.$$

se tiene que

$$\langle \phi(x_i) \cdot \phi(x_j) \rangle = \sum_{t=1}^{n} \lambda_t v_{ti} v_{tj} = (\mathbf{V}\mathbf{\Lambda}\mathbf{V}')_{ij} = \mathbf{K}_{ij} = \mathbf{K}(\mathbf{x_i} \cdot \mathbf{x_j})$$

implica que $K(\mathbf{x}, \mathbf{z})$ es un función kernel correspondiente al mapeo ϕ. El requerimiento de que los eigenvalores de $\mathbf{K}$ sean no negativos es necesario, ya que si tenemos un eigenvalor negativo λ_s en el eigenvector $\mathbf{v}_s$, el punto

$$\mathbf{z} = \sum_{i=1}^{n} v_{si} \phi(\mathbf{x}_i) = \sqrt{\mathbf{\Lambda}}\mathbf{V}'\mathbf{v}_s$$

en el espacio de características podría tener norma cuadrada

$$\|\mathbf{z}\|^2 = \langle \mathbf{z} \cdot \mathbf{z} \rangle = \mathbf{v}_s' \mathbf{V} \sqrt{\mathbf{\Lambda}} \sqrt{\mathbf{\Lambda}} \mathbf{V}' \mathbf{v}_s = \mathbf{v}_s' \mathbf{V} \mathbf{\Lambda} \mathbf{V}' \mathbf{v}_s = \mathbf{v}_s' \mathbf{K} \mathbf{v}_s = \lambda_s < 0,$$

contradiciendo la geometría de este espacio. Esto nos lleva a la siguiente proposición

Proposición 2.3 *Sea X un espacio de entrada finito con una función simétrica sobre X $K(\mathbf{x}, \mathbf{z})$.Decimos que $K(\mathbf{x}, \mathbf{z})$ es una función kernel si y solamente si la matriz*

$$\mathbf{K} = (K(\mathbf{x}_i \cdot \mathbf{x}_j))_{i,j=1}^n$$

es positiva semidefinida (tiene eigenvalores no negativos)

Permitiendo una ligera generalización de un producto punto en un espacio de Hilbert [15] introduciendo un peso λ_i para cada dimensión

$$\langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \cdot \phi_i(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}),$$

por lo tanto, el vector de características sería

$$\boldsymbol{\phi}(\mathbf{x}) = (\phi_i(\mathbf{x}), \phi_i(\mathbf{x}), \ldots, \phi_i(\mathbf{x}), \ldots)\,.$$

El teorema de Mercer, da las condiciones necesarias y suficientes para que una función simétrica continua $K(\mathbf{x}, \mathbf{z})$ sea representada

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \cdot \phi_i(\mathbf{z})$$

con λ_i no negativos, que es equivalente a que $K(\mathbf{x}, \mathbf{z})$ sea un producto punto en el espacio de características $F \supseteq \boldsymbol{\phi}(\mathbf{X})$, donde F es el espacio l_2 de todas las secuencias

$$\boldsymbol{\psi} = (\psi_1, \psi_2, \ldots \psi_i, ...).$$

para el cual

$$\sum_{i=1}^{\infty} \lambda_i \psi_i^2 < \infty.$$

Esto implícitamente induce un espacio definido por el vector de características y como una consecuencia una función lineal en F puede ser representada por

$$f(x) = \sum_{i=1}^{\infty} \lambda_i \psi_i \phi_i(\mathbf{x}) + b = \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}, \mathbf{x}_j) + b$$

donde la primera expresión es la representación original y la segunda es el dual [20]. La relación entre los dos está dada por

$$\boldsymbol{\psi} = \sum_{j=1}^{l} \alpha_j y_j \boldsymbol{\phi}(\mathbf{x}_j).$$

En la representación original, el número de términos en la suma es igual a la dimensionalidad en el espacio de características, mientras que en el dual existen l términos. La analogía con el caso finito es similar. La contribución a partir del análisis funcional nos conduce al problema para ecuaciones integrales de la forma

$$\int_X K(\mathbf{x}, \mathbf{z}) \phi(\mathbf{z}) d\mathbf{z} = \boldsymbol{\lambda} \phi(\mathbf{x})$$

donde $K(\mathbf{x}, \mathbf{z})$ es una función kernel acotada, simétrica y positiva y X es un espacio compacto.

Teorema 2.2 *(Mercer) Sea X un subconjunto compacto de $\mathbb{R}^n$. Suponiendo que $\mathbf{K}$ es una función simétrica continua tal que el operador integral $T_K : L_2(X) \to L_2(x)$,*

$$(T_K f)(\cdot) = \int_X K(\cdot, \mathbf{x}) f(\mathbf{x}) d\mathbf{x},$$

es positivo, esto es

$$\int_{X \times X} K(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mathbf{x}d\mathbf{z} \geq \mathbf{0},$$

para toda $f \in L_2(X)$. Entonces $K(\mathbf{x}, \mathbf{z})$ puede ser expandida en una serie uniformemente convergente (sobre $X \times X$) en términos de las eigenfunciones $\phi_j \in L_2(X)$, normalizadas de tal forma que $\left\|\phi_j\right\|_{L_2} = 1$ y eigenvalores asociados positivos $\lambda_j \geq 0$,

$$K(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{z}).$$

2.2.6. Caso linealmente no separable

Los clasificadores lineales presentados en las dos secciones anteriores son muy limitados. En la mayoría de las clases, no únicamente se traslapan o intersectan los datos al generar un hiperplano de separación, sino que la separación genuina de estos datos está dada por hiper-superficies no-lineales. Una característica del enfoque presentado anteriormente radica en que éste, puede ser fácilmente extendido para crear cotas de decisión no lineal. El motivo de tal extensión es que una SVM puede crear una hipersuperficie de decisión no lineal, capaz de clasificar datos separables no linealmente. Generalmente, para patrones de entrada n-dimensionales, en lugar de una curva no lineal, una SVM creará una hipersuperficie de separación no lineal.

El problema de optimización utilizando kernels queda de la siguiente manera [20] [92]:

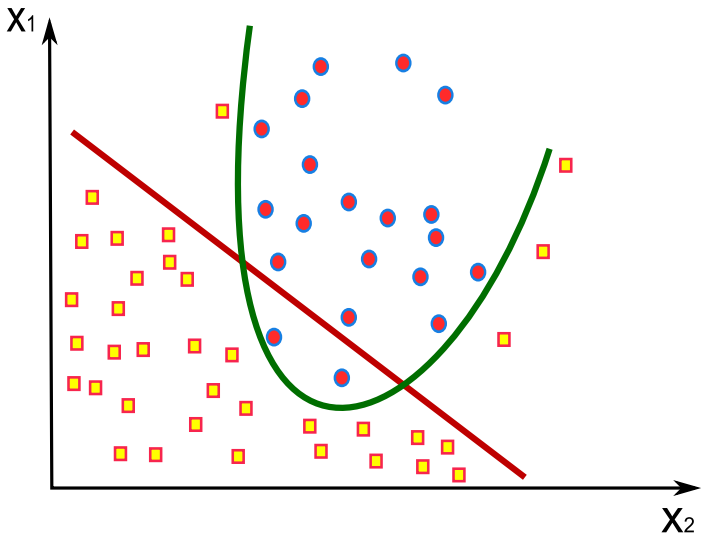

Figura 2.5: Clasificador No-Lineal

Proposición 2.4 *Dado un conjunto de datos de entrenamiento* $S = [(x_1, y_1) \cdots (x_l, y_l)]$, *un espacio de características* $\phi(x)$ *definido por el kernel* $K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle$, *la solución de*

$$\max_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \left[K(x_i, x_j) + \frac{1}{C} \delta_{ij} \right] + \sum_{i=1}^{l} \alpha_i$$
$$\text{sujeto a : } \sum_{i=1}^{l} \alpha_i y_i = 0$$

es α_i^*, $f(x) = \sum_{i=1}^{l} \alpha_i^* y_i K(x_i, x) + b^*$ *,donde* b^* *es elegido tal que*

$$y_i f(x_i) = 1 - \xi^* = 1 - \frac{\alpha^*}{C}$$
$$w^* = \sum_{i=1}^{l} \alpha_i^* y_i K(x, x),$$

la regla de decisión $sgn[f(x)]$ *es equivalente al hiperplano en el espacio de características definido por el kernel* $K(x,z)$ *el cual resuelve el problema de optimización. Luego, el margen geométrico está dado por*

$$\gamma^* = \left(\sum_{i \in sv} \alpha_i^* - \frac{1}{C} \langle \alpha^* \cdot \alpha^* \rangle \right)^{-\frac{1}{2}}$$

Utilizando el Kernel

$$K'(x,z) = K(x,z) + \frac{1}{C}\delta_x(z)$$

El margen blando en L1

$$\begin{gathered} \min_{w,b,\xi_i} \langle w \cdot w \rangle + C \sum_{i=1}^{l} \xi_i \\ \text{sujeto a: } y_i \left(\langle w \cdot x_i \rangle + b \right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{gathered} \tag{2.10}$$

Donde el Lagrangiano original es

$$L(w,b,\xi_i,\alpha) = \frac{1}{2}\langle w \cdot w \rangle - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\langle w \cdot x_i \rangle + b \right) - 1 + \xi_i \right] + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \gamma_i \xi_i$$

De donde tenemos que el dual está dado por

$$w(\alpha) = -\frac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle x_i \cdot x_j \rangle + \sum_{i=1}^{l} \alpha_i$$

Este es el mismo que el margen máximo, pero

$$C - \alpha_i - \gamma_i = 0, \gamma_i \geq 0 \Rightarrow C \geq \alpha_i$$

Con las condiciones de Kuhn-Tucker

$$\begin{gathered} \gamma_i \xi_i = 0 \text{ ó } (\alpha_i - C)\xi_i = 0 \\ \alpha_i \left[y_i \left(\langle w \cdot x_i \rangle + b \right) - 1 + \xi_i \right] = 0 \end{gathered}$$

donde $\xi_i \neq 0$, $\gamma_i = 0, \Rightarrow C = \alpha_i$, con $\xi_i = 0$ el margen es máximo, α_i es positivo y puede incrementarse hasta C, por lo tanto $C \geq \alpha_i \geq 0$

Proposición 2.5 *Dado un conjunto de datos de entrenamiento* $S = [(x_1, y_1) \cdots (x_l, y_l)]$, *un espacio de características* $\phi(x)$ *definido por el kernel* $K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle$, *la solución de*

$$\begin{gathered} \max_{\alpha_i} -\tfrac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j K(x_i, x_j) + \sum_{i=1}^{l} \alpha_i \\ \textit{sujeto a: } \sum_{i=1}^{l} \alpha_i y_i = 0, \ C \geq \alpha_i \geq 0 \end{gathered}$$

es α_i^*, $f(x) = \sum_{i=1}^{l} \alpha_i^* y_i K(x_i, x) + b^*$, *la regla de decisión* $sgn[f(x)]$ *es equivalente al hiperplano en el espacio de características definido por el Kernel* $K(x, z)$, *el cual resuelve el problema de optimización. El margen geométrico está dado por*

$$\gamma^* = \left(\sum_{i \in sv} \alpha_i^* \right)^{-\frac{1}{2}}$$

Cuando la cota de α_i es C, se origina el problema del máximo margen.

Elegir C es lo mismo que obtener v en

$$\begin{gathered} \max_{\alpha_i} -\tfrac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j K(x_i, x_j) \\ \text{sujeto a: } \sum_{i=1}^{l} \alpha_i y_i = 0, \\ \sum_{i=1}^{l} \alpha_i \geq v, \\ \tfrac{1}{l} \geq \alpha_i \geq 0 \end{gathered}$$

Para una solución no-*óptima*, $\hat{\alpha}$ es el valor actual de las variables duales. El vector de pesos es calculado por $\frac{\partial L}{\partial w} = 0$. La solución $\hat{w} \cdot \hat{w}$ satisface las condiciones originales con $\frac{1}{2}\|\hat{w}\|^2 + c\sum_{i=1}^{l} \xi_i$, $\inf_{w,b} L(w, b, \hat{\alpha})$ como solución dual factible. Ya que:

$$\begin{aligned} L &= \frac{1}{2} \langle w \cdot w \rangle - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\langle w \cdot x_i \rangle + b \right) - 1 \right], \\ w &= \sum_{i=1}^{l} \hat{\alpha}_i y_i x_i, \\ \frac{\partial L}{\partial b} &= 0 \rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned}$$

calculamos la diferencia entre las soluciones original y dual factibles $C - \alpha_i = \gamma_i$

$$\begin{aligned} &\tfrac{1}{2}\|w\|^2 + c\sum_{i=1}^{l} \xi_i - \inf_{w,b} L(w, b, \hat{\alpha}) \\ &= \sum_{i=1}^{l} \hat{\alpha}_i \left[y_i \left(\sum_{j=1}^{l} y_j \alpha_j K \langle x_j \cdot x_i \rangle + b \right) - 1 + \xi_i \right] + \sum_{i=1}^{l} \gamma_i \xi_i \\ &= C \sum_{i=1}^{l} \xi_i + \sum_{i=1}^{l} \hat{\alpha}_i \left[y_i \left(\sum_{j=1}^{l} y_j \alpha_j K \langle x_j \cdot x_i \rangle + b \right) - 1 \right] \\ &= \sum_{i,j=1}^{l} \hat{\alpha}_i y_i y_j \alpha_j K \langle x_j \cdot x_i \rangle - \sum_{i=1}^{l} \hat{\alpha}_i + C \sum_{i=1}^{l} \xi_i \\ &= \sum_{i=1}^{l} \hat{\alpha}_i - 2w(\alpha) + C \sum_{i=1}^{l} \xi_i \end{aligned}$$

2.3. Implementaciones de SVM para grandes conjuntos de datos

En esta Sección se muestran algunas de las implementaciones de SVM más empleadas actualmente como son Optimización mínima secuencial (SMO), LibSVM y Simple Support Vector Machines (SSVM).

2.3.1. *SMO*

El algoritmo de Optimización Mínima Secuencial [67] es obtenido a partir de la idea del método de descomposición a su extremo, al optimizar un subconjunto mínimo de únicamente dos puntos en cada iteración. El poder de esta técnica reside en el hecho de que el problema de optimización para dos puntos admite una solución analítica, eliminando la necesidad de usar un optimizador de programación cuadrática iterativo como parte del algoritmo [36].

El requisito de que la condición $\sum_{i=1}^{l} \alpha_i y_i = 0$ obliga en todo momento a que el número de multiplicadores que puede ser optimizado en cada paso es 2. Cada vez que un multiplicador es actualizado, por lo menos otro multiplicador necesita ser ajustado con el propósito de mantener la condición verdadera. En cada paso, SMO elige dos elementos α_i y α_j para optimizarlos, encuentra el valor óptimo de esos dos parámetros, dado que los demás se encuentran fijos y actualiza el vector α. La elección de los dos puntos es determinada por una heurística, mientras que la optimización de los dos multiplicadores es realizada analíticamente.

Experimentalmente, el desempeño de SMO es muy bueno para SVM con entradas escasas, así como para SVM no lineales. Esto es debido a que el tiempo de computación del kernel puede ser reducido, mejorando directamente su desempeño. A pesar de que necesita más iteraciones para converger, cada iteración usa solo pocas operaciones, por lo tanto converge muy rápido. Además del tiempo de convergencia, otra característica del algoritmo radica en que este no necesita almacenar la matriz del kernel en la memoria, ya

que no se involucran operaciones matriciales. El algoritmo SMO se desempeña bien para problemas grandes, porque éste escala bien con el tamaño del conjunto de entrenamiento. Los autores de SMO aseguran que es un fuerte candidato para llegar a ser el algoritmo de entrenamiento estándar de SVM [67].

2.3.2. *Libsvm*

La mayoría de los métodos de descomposición obtienen un conjunto de datos inicial a partir del conjunto de datos entero, este conjunto de datos es optimizado en cada iteración, mejorando el valor de la función objetivo en cada iteración. El proceso iterativo llega a su fin cuando un criterio de paro derivado de las condiciones de Karush-Kuhn-Tucker es satisfecho o una precisión requerida es alcanzada. El algoritmo LIBSVM está basado en el algoritmo SMO, sin embargo, posee un algoritmo de selección del conjunto de trabajo mucho más avanzado. LIBSVM emplea un algoritmo de dirección de búsqueda que maximiza el incremento la función objetivo en cada iteración. El algoritmo inicia realizando una primera aproximación de la función objetivo obteniendo un vector α. A partir de esta primera aproximación calcula $\alpha' = \alpha + \lambda u$, donde la dirección u tiene únicamente dos coeficientes no cero. El algoritmo emplea dos direcciones de búsqueda, una dirección de búsqueda u^{ij} para λ positivos y una dirección de búsqueda $-u^{ij} = u^{ji}$ para λ negativo. La dirección de búsqueda más efectiva para cada iteración será la dirección que maximice el incremento de la función objetivo.

2.3.3. *SSVM*

El algoritmo *Simple Support Vector Machines* fue propuesto por Vishwanathan y Murty [96], el algoritmo inicializa con un pequeño conjunto de datos, este conjunto es obtenido mediante un algoritmo que detecta y obtiene los datos más cercanos entre clases opuestas, a este conjunto lo llaman conjunto candidato de vectores soporte. Una vez que el algoritmo encuentra un punto que viola las condiciones en el conjunto de datos, el algoritmo lo aparta y adiciona al conjunto candidato de vectores soporte, los datos que

se encuentran ya presentes en el conjunto candidato son eliminados, con el objetivo de asegurar que las condiciones de KKT se cumplan, este procedimiento es repetido en todo el conjunto de datos hasta que todos los puntos han sido evaluados y ningún otro punto que viole las condiciones ha sido encontrado.

Capítulo 3

Clasificación de SVM en Dos Etapas ($\mathbf{SVM^2}$)

Vivimos en un mundo lleno de datos. Día con día, la gente se encuentra manejando grandes cantidades de información y almacenándola o representándola como datos para su posterior análisis y empleo. Una de las prácticas más comunes al manejar conjuntos de datos es clasificarlos o agruparlos dentro de un conjunto de categorías o grupos. La clasificación juega un rol indispensable e importante a lo largo del desarrollo de la historia de la humanidad. Con el objetivo de aprender un nuevo objeto o entender un nuevo fenómeno, la gente siempre trata de ver las características que puede describir o comparar con otros objetos o fenómenos conocidos, basándose en semejanzas o disimilaridades, con el objetivo de generalizar de acuerdo a algún patrón o determinada regla.

Una de las técnicas de clasificación más empleadas en los últimos años para clasificación de datos y regresión son las SVM. Las SVM son una poderosa técnica empleada en clasificación de datos y análisis de regresión. Debido a sus buenos fundamentos teóricos y su buena capacidad de generalización las SVM han llegado a ser en los últimos años, uno de los métodos de clasificación más utilizado. En este Capítulo, presentamos tres nuevas técnicas de clasificación con SVM para conjuntos de datos grandes. Las técnicas presentadas emplean métodos de agrupamiento con el objetivo de eliminar datos

no representativos del conjunto de datos y agilizar el entrenamiento de las SVM.

3.1. Introducción

Durante la fase de entrenamiento de un clasificador, los algoritmos existentes generalmente tratan de maximizar el desempeño de clasificación para los datos de entrenamiento. Sin embargo, si el clasificador es acondicionado demasiado para el conjunto de datos de entrenamiento, la habilidad de clasificación para datos desconocidos, i.e., la habilidad de generalización es degradada. Éste fenómeno es conocido como sobreentrenamiento u *overfitting*. Las SVM han recibido una gran atención debido a que optimizan la solución esperada. Las SVM obtienen cotas de decisión a partir de los datos de entrenamiento, de tal forma que la separación o margen existente entre las fronteras de cada clase sea maximizada en un espacio altamente dimensional, i.e. cuando un conjunto de datos de entrada es linealmente no separable en un espacio d, éste es enviado a un espacio m dimensional ($m > d$). La estrategia empleada por las SVM minimiza los errores de clasificación de los datos de entrenamiento y optimiza su capacidad de generalización. Una de las ventajas más grandes de las SVM radica en que uno puede obtener el mismo hiperplano de separación si borramos los datos que satisfacen estrictamente las condiciones de desigualdad de la función objetivo. Los datos que satisfacen las condiciones de igualdad son llamados *vectores soporte*. Esta ventaja es explotada en este Capítulo con el objetivo de reducir el conjunto de datos de entrada.

Sin embargo, aunque las SVM tienen un buen desempeño al generalizar, el tiempo de aprendizaje de éstas suele ser cuadrático con respecto al conjunto de datos de entrada, presentado problemas al trabajar con grandes conjuntos de datos de entrada. Siendo la principal desventaja de las SVM el tiempo de entrenamiento con grandes conjuntos de datos, en este Capítulo se presentan tres nuevos algoritmos de clasificación para grandes conjuntos de datos basados en SVM. Los algoritmos propuestos reducen significativamente el tiempo de entrenamiento llevado a cabo por otras implementaciones de SVM diseñadas para trabajar con grandes conjuntos de datos.

3.1.1. Clasificación con SVM

Un gran número de implementaciones basadas en SVM han sido desarrolladas con el objetivo de afrontar el problema de clasificación con SVM para grandes conjuntos de datos. Generalmente estos métodos pueden ser divididos en dos tipos:

1. algoritmos de búsqueda de candidatos a vectores soporte.

2. algoritmos de descomposición del conjunto de datos de entrada, descomponiendo la matriz de entrada Q en pequeñas matrices.

Diversos algoritmos han sido propuestos en la literatura para agilizar el tiempo de entrenamiento de las SVM. Dos de las técnicas más conocidas son descomposición o *Chunking* [13] [67] y SMO [67]. El algoritmo de descomposición o *Chunking* emplea un subconjunto arbitrario de datos llamado "*chunk*" y entrena una SVM sobre esta porción de datos. En cada iteración, el algoritmo selecciona un conjunto de k puntos que viola las condiciones de Karush-Kuhn-Tucker. Estos k puntos son adheridos a los vectores soporte del *chunk* anterior para formar un nuevo *chunk*. La iteración es realizada hasta que se satisface un criterio de paro. El algoritmo SMO es obtenido a partir de la idea de descomponer un conjunto de datos a su extremo y optimizar un subconjunto mínimo de únicamente dos puntos en cada iteración. El tiempo de entrenamiento es reducido significativamente debido a que el problema de optimización para dos puntos admite una solución analítica, eliminando la necesidad de usar un optimizador de programación cuadrática iterativo como parte del algoritmo [13][36][81].

En [52] Li et al, proponen un enfoque basado en una técnica de aprendizaje incremental y múltiples aproximaciones empleando SVM. Algunas otras técnicas incluyen Selección Aleatoria [5] [79] [88] y Rocchio Bundling [81]; éstas muestrean un pequeño número de datos de entrenamiento a partir del conjunto de datos original de tal forma que sea maximizado el grado de aprendizaje. Sin embargo, los algoritmos de muestreo podrían simplificar el conjunto de datos de entrenamiento perdiendo los beneficios de usar SVM, especialmente cuando la distribución de probabilidad de datos de entrenamiento y prueba son diferentes.

Algunos autores han propuesto segmentar el conjunto de entrenamiento en grupos [19][99]. Awad et al. describen un algoritmo para aproximar los vectores soporte aplicando análisis de agrupamiento jerárquico, usando para ello una estructura jerárquica producida por el algoritmo DGSOT [2]. Sin embargo, construir la estructura jerárquica de muchos conjuntos de entrenamiento implementa un alto costo computacional. Por otro lado, en agrupamiento particional, el número de grupos es predefinido y determinar el número óptimo de grupos puede involucrar un costo computacional mucho más alto que el de agrupamiento.

El agrupamiento es un procedimiento de clasificación no supervisada, que tiene como objetivo clasificar objetos similares dentro de diferentes grupos i.e., particionar el conjunto de datos dentro de subconjuntos de grupos, de tal forma que los datos en cada subconjunto compartan características comunes (a menudo proximidad de acuerdo a alguna medida de distancia definida) [74]. El objetivo de agrupamiento es separar un conjunto de datos finito no etiquetado dentro de un conjunto finito y discreto de estructuras de datos ocultas "naturales" [74]. Algunos resultados [2][19][99] muestran que las técnicas de agrupamiento pueden ayudar a decrementar la complejidad de entrenamiento de las SVM. Sin embargo, cuando éstas trabajan con grandes conjuntos de datos, el tiempo de cómputo también es muy grande debido a que necesitan construir la estructura jerárquica.

Otras técnicas interesantes usadas para entrenar SVM con grandes conjuntos de datos emplean agrupamiento jerárquico [99] escaneando el conjunto de datos entero una sola vez. Los autores aseguran que el método propuesto escala bien para grandes conjuntos de datos y las precisiones obtenidas mediante este método son comparables a las otras implementaciones de SVM. Sin embargo, el micro-agrupamiento empleado está restringido a trabajar con únicamente kernels lineales. Saketha et al. [77] Proponen un algoritmo basado en BIRCH con el objetivo de estimar las estadísticas de los componentes. El método propuesto es escalable para grandes conjuntos de datos. Sin embargo, el algoritmo muestrea el conjunto de datos de entrada, cuyo proceso podría afectar el proceso de entrenamiento de las SVM, especialmente cuando la probabilidad de distribución de los

datos de entrenamiento y prueba son diferentes [99]. Khan et al. [42] aplican técnicas de reducción mediante agrupamiento, el algoritmo trata de encontrar vectores soporte relevantes y agilizar el proceso de entrenamiento al construir un árbol de agrupamiento jerárquico para cada clase a partir del conjunto de datos de entrada; esto lo hace de forma iterativa en varias etapas de entrenamiento. En cada etapa, una SVM es entrenada sobre los nodos de cada árbol. Los vectores soporte del clasificador determinan el crecimiento del árbol, mientras que los puntos que no son vectores soporte son eliminados. Este método es escalable para grandes conjuntos de datos, sin embargo el algoritmo es sensible en conjuntos de datos con ruido y susceptible a conjuntos de datos incompletos. Chen at al. [11] emplean múltiples clasificadores para reducir el tiempo de entrenamiento de una SVM. Los autores desarrollan un enfoque mediante agrupamiento *K-Means* con el objetivo de seleccionar y combinar un número dado de clasificadores tomando en cuenta precisión y eficiencia. Las precisiones obtenidas empleando este algoritmo son buenas, sin embargo este método trabaja con conjuntos de datos de pequeño y mediano tamaño.

3.1.2. Estrategia propuesta

De acuerdo a la arquitectura de las SVM, la solución está dada por un pequeño conjunto de datos llamado *vectores soporte* que definen los límites entre una clase y otra, por lo cual, únicamente los datos cercanos entre cada frontera son necesarios. Por otro lado, ya que el tiempo de entrenamiento de las SVM llega a ser enorme cuando el número de datos de entrada incrementa, el tiempo de entrenamiento puede ser reducido considerablemente si los datos alejados de la frontera de clasificación son eliminados. Por lo tanto, si podemos eliminar de una forma eficiente los datos alejados del hiperplano de clasificación antes del entrenamiento, podemos agilizar el proceso de entrenamiento. Sin embargo, no es posible conocer de antemano los datos cercanos a la frontera de decisión sin aplicar alguna técnica de búsqueda.

La estrategia que proponemos, emplea dos fases de clasificación con SVM. Los pasos

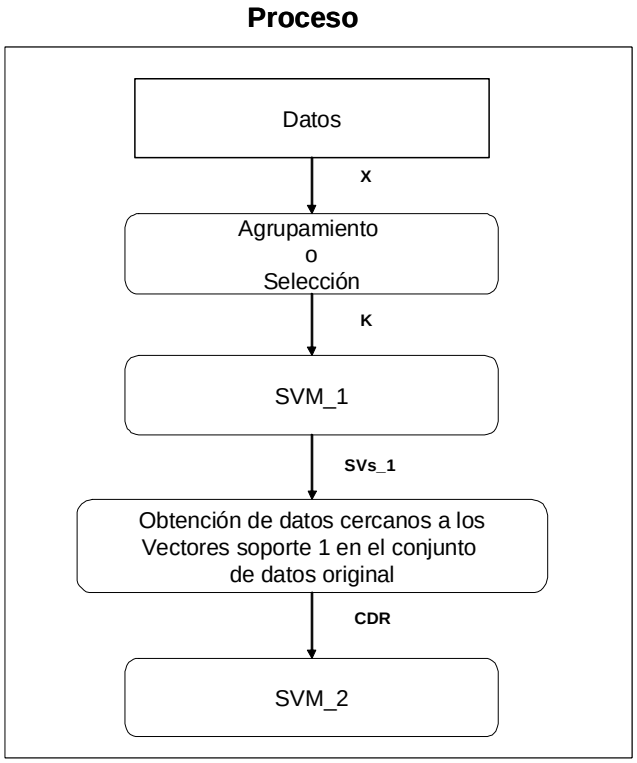

Figura 3.1: Estrategia de clasificación con SVM. En la Figura **X** es el conjunto de datos original, **K** representa los centros de grupos o datos seleccionados aleatoriamente, **SV1** son los vectores soporte obtenidos en la primera fase de SVM, **CDR** el conjunto de datos reducido.

llevados a cabo por el algoritmo son descritos a continuación:

1. primero, los algoritmos propuestos agrupan el conjunto de datos de entrada o seleccionan un conjunto de datos aleatorio (ver Figura 3.1 -agrupamiento o selección-) a partir del conjunto de datos de entrada,

2. la siguiente fase es emplear una primera etapa de SVM (ver Figura 3.1 -SVM1-) sobre el conjunto de centros de grupos o datos seleccionados aleatoriamente, el objetivo es encontrar los datos mas representativos (vectores soporte) de este conjunto de datos. Aún cuando el hiperplano obtenido es únicamente un esbozo del hiperplano real, ya que fue obtenido a partir de un conjunto de datos reducido, éste es de gran utilidad para seleccionar y recuperar todos los datos más importantes del conjunto de datos original.

3. en ésta fase se recuperan todos los datos cercanos a SV1 en el conjunto de datos original (ver Figura 3.1-Obtención de datos cercanos a VS-) , la finalidad es mejorar el primer hiperplano obtenido.

4. una vez obtenido el conjunto de datos reducido, se emplea una segunda etapa de clasificación con SVM, ésta segunda etapa refina el hiperplano obtenido inicialmente y mejora su capacidad de generalización (ver Figura 3.1 -SVM2-).

En esta sección, proponemos tres enfoques para clasificación de grandes conjuntos de datos llamados: *clasificación de SVM en dos etapas basado en agrupamiento difuso* (FCM-SVM^2), *clasificación de SVM en dos etapas basado en esferas de cerradura mínima* (MEB-SVM^2) y *clasificación de SVM en dos etapas basado en selección aleatoria*(RS-SVM^2).

El primer enfoque propuesto (FCM-SVM^2) particiona el conjunto de datos de entrada en un conjunto de grupos predefinido. Después de particionar el conjunto de datos de entrada, una primera etapa de SVM es realizada sobre los centros de los grupos con el objetivo de aproximar un hiperplano de clasificación inicial y eliminar aquellos puntos

que son irrelevantes (aquellos puntos que se encuentran más alejados del hiperplano de clasificación). Determinado el primer hiperplano de separación, eliminamos aquellos grupos que se encuentran más alejados del hiperplano. Al conjunto de grupos restante, aplicamos un proceso de desagrupamiento y obtenemos un conjunto de datos reducido. Una vez obtenido un conjunto de datos reducido, utilizamos por último SVM para extraer los vectores soporte y refinar el hiperplano de clasificación inicial obtenido.

El segundo enfoque propuesto (MEB-SVM^2), está basado en agrupamiento mediante Esferas de Cerradura Mínima (*Minimum Enclosing Ball* -MEB-, por sus siglas en ingles)[45] y de ahí es llamado *Clasificación de SVM basado en Esferas de Cerradura Mínima (*MEB-SVM^2*).* Las MEB son utilizadas en este Capítulo como un medio para particionar el espacio de datos de entrada, agrupando totalmente el conjunto de datos de entrada y obteniendo un conjunto de centros de grupos con sus respectivos radios. El algoritmo MEB calcula la esfera de radio más pequeña que encierra un conjunto de datos de puntos dados. Los algoritmos tradicionales para encontrar MEB exactas escalan bien al aumentar el número de dimensiones del conjunto de datos de entrada. En esta sección se emplea el enfoque propuesto en [45] para encontrar un conjunto de MEBs a partir del conjunto de datos de entrada que seccionen completamente el conjunto de datos de entrada. Una propiedad de este enfoque radica en el hecho de que una MEB es independiente del tamaño y la dimensión del conjunto de datos de entrada. Una vez obtenidos sus centros se utilizó SVM sobre los centros de las esferas, generando un hiperplano de separación. Determinado el hiperplano de separación se eliminan aquellas esferas que se encuentren más alejadas del hiperplano de separación. Después de los pasos previamente descritos, se obtiene el conjunto de datos más importante, que no incluye aquellos puntos más alejados del hiperplano de clasificación. A partir del conjunto de esferas restante, se aplica un proceso de desagrupamiento y se obtiene un conjunto de datos reducido que posee las características más importantes del conjunto de datos original. Una vez obtenido el conjunto de datos reducido, se utiliza por último SMO para extraer los vectores soporte. En los resultados experimentales, se muestra que la precisión obtenida mediante MEB-SVM^2 es muy cercana a la obtenida mediante otros métodos, con la

diferencia de que el tiempo de aprendizaje empleado por MEB-SVM^2 es mucho menor.

El tercer enfoque propuesto (RS-SVM^2), se selecciona un conjunto de datos iniciales de forma aleatoria, después de la primera esta fase obtenemos un hiperplano de clasificación inicial utilizando SVM, los demás pasos son idénticos al segundo enfoque propuesto. Seleccionar de forma aleatoria el conjunto de datos reduce el tiempo inicial de seccionamiento del conjunto de datos de entrada. En los resultados experimentales, se muestra que la precisión obtenida mediante RS-SVM^2 es muy cercana y en algunos casos es igual a la obtenida mediante MEB-SVM^2 , con la diferencia de que el tiempo de aprendizaje empleado es aún menor.

3.2. SVM en dos etapas basado en FCM (FCM-SVM^2)

En esta sección se propone un algoritmo para agilizar el tiempo de entrenamiento de las SVM. Algunos algoritmos basados en lógica difusa han sido propuestos en la literatura [17] [33] [18]. Bao et al. proponen un algoritmo de regresión basado en SVM y lógica difusa, el algoritmo es diseñado para predicciones en la bolsa [17]. Hong y Hwang, proponen un algoritmo de regresión basado en SVM y lógica difusa, el algoritmo obtiene una buena eficiencia computacional [33]. Bhattacharya et al. emplean SVM y Fuzzy C-Means para aumentar la precisión en la recuperación y representación de imágenes [18]. El algoritmo propuesto en esta sección funciona empleando agrupamiento particional. El algoritmo secciona el conjunto de datos de entrada empleando un algoritmo de agrupamiento y una primera etapa de SVM con el objetivo de aproximar un hiperplano de clasificación inicial y eliminar aquellos puntos que son irrelevantes (aquellos puntos que se encuentran más alejados del hiperplano de clasificación). Elegir el número óptimo de grupos en los que se debe dividir un conjunto de datos puede requerir un costo computacional muy alto. En este caso, es conveniente que el número de grupos sea grande. De hecho, no necesitamos encontrar el óptimo, sino elegir un número grande

de grupos, que depende directamente del tamaño original del conjunto de datos. En los experimentos realizados se utilizó como máximo un 0.1 % del conjunto total de datos. Con el propósito de manejar grandes cantidades de datos utilizamos agrupamiento particional y dividimos el conjunto original de datos en un número predefinido de grupos. Una vez obtenidos los centros de los grupos utilizamos SVM sobre ellos generando un hiperplano de separación. Por esta razón, el tiempo de aprendizaje es muy pequeño, ya que el tiempo de entrenamiento depende del número de grupos y no directamente del número total de registros de datos. Determinado el primer hiperplano de separación, eliminamos aquellos grupos que se encuentran más alejados del hiperplano. Al conjunto de grupos restante, aplicamos un proceso de desagrupamiento y obtenemos un conjunto de datos reducido. Una vez obtenido un conjunto de datos reducido, utilizamos por último SVM para extraer los vectores soporte y refinar el hiperplano de clasificación inicial obtenido. En los resultados experimentales se muestra que el conjunto de vectores soporte obtenido mediante Agrupamiento+SVM es muy cercano al conjunto de vectores obtenido mediante SVM con la diferencia de que el tiempo de aprendizaje empleado por FCM-SVM^2 es mucho menor.

El agrupamiento esencialmente realiza la tarea de seccionar un conjunto de patrones dentro de un conjunto más o menos homogéneo de clases (grupos) con respecto a una medida de similaridad dada, de tal forma que los patrones correspondientes a un grupo sean similares entre sí y los patrones de diferentes grupos sean tan diferentes como sea posible.

3.2.1. Fuzzy C-Means

Con el objetivo de formular el problema de agrupamiento difuso, consideremos un conjunto finito de elementos $X = \{x_1, x_2, \ldots, x_n\}$ con dimensión d en el espacio euclidiano $\mathbb{R}^d$, i.e., $x_j \in \mathbb{R}^d, j = 1, 2, \ldots, n$. El algoritmo de agrupamiento deberá particionar los datos de entrada dentro de k conjuntos difusos con respecto a un criterio dado. El criterio usualmente radica en optimizar una función objetivo. El resultado del agrupamiento

difuso puede ser expresado por una matriz de partición U tal que $U = [u_{ij}]_{i=1\ldots c,\ j=1\ldots n}$, donde u_{ij} es un valor numérico en $[0, 1]$. Existen dos condiciones sobre el valor de u_{ij}. Primero, las funciones de membresía totales de los elementos $x_j \in X$ en todas las clases es igual a 1. Segundo, cada grupo construido es no vacío y diferente del conjunto entero, i.e.,

$$\begin{aligned} &\sum_{i=1}^{c} u_{ij} = 1, \quad \text{para toda } j = 1, 2, \ldots, n \\ &0 < \sum_{j=1}^{n} u_{ij} < n, \quad \text{para toda } i = 1, 2, \ldots, c. \end{aligned} \tag{3.1}$$

Una forma general de la función objetivo [32] es

$$J\left(u_{ij}, \mathbf{v}_k\right) = \sum_{i=1}^{c} \sum_{j=1}^{n} \sum_{k=1}^{c} g\left[w(x_i), u_{ij}\right]\ d\left(\mathbf{x}_j, \mathbf{v}_k\right)$$

donde $w(x_i)$ es el peso a-priori para cada $\mathbf{x}_i$, $d\left(\mathbf{x}_j, \mathbf{v}_i\right)$ es el grado de disimilaridad entre el dato $\mathbf{x}_j$ y el elemento suplemental $\mathbf{v}_k$, que puede ser considerado como el vector central del i-ésimo grupo. El grado de disimilaridad es definido como la medida que satisface dos axiomas: 1) $d\left(\mathbf{x}_j, \mathbf{v}_k\right) \geq 0$, 2) $d\left(\mathbf{x}_j, \mathbf{v}_k\right) = d\left(\mathbf{x}_k \mathbf{v}_j\right)$. El agrupamiento difuso puede ser formulado dentro de un problema de optimización:

$$\begin{aligned} \text{Min} \quad J\left(u_{ij}, \mathbf{v}_k\right),\ i, k \quad &= \quad 1, 2 \ldots c;\ j = 1, 2 \ldots n \\ \text{Sujeto a} \quad &: \quad (3.1) \end{aligned}$$

donde la función objetivo es

$$J\left(u_{ij}, \mathbf{v}_k\right) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m} \left\|\mathbf{x}_j - \mathbf{v}_i\right\|^2, \qquad m > 1 \tag{3.2}$$

donde m es llamado un peso exponencial que influye en el grado de difusividad de la matriz de membresía. Para resolver el problema de minimización, diferenciamos la función objetivo en la ecuación (3.2) con respecto a $\mathbf{v}_i$ (para u_{ij} fijo, $i = 1, \ldots, c$, $j = 1, \ldots, n$) y para u_{ij} (para $\mathbf{v}_i$ fijo, $i = 1, \ldots, c$) y aplicamos las condiciones de la

ecuación (3.1), obteniendo

$$v_i = \frac{1}{\sum_{j=1}^{n} (u_{ij})^m} \sum_{j=1}^{n} (u_{ij})^m \mathbf{x}_j, \quad i = 1, \ldots, c \tag{3.3}$$

$$u_{ij} = \frac{\left(1/\left\|\mathbf{x}_j - \mathbf{v}_i\right\|^2\right)^{1/m-1}}{\sum_{k=1}^{c} \left(1/\left\|\mathbf{x}_j - \mathbf{v}_k\right\|^2\right)^{1/m-1}} \tag{3.4}$$

donde $i = 1, \ldots, c$; $j = 1, \ldots, n$. El sistema descrito por las ecuaciones (3.3) y (3.4) no puede ser resuelto analíticamente. Sin embargo, el algoritmo FCM provee un enfoque iterativo, el cual es resumido de la siguiente forma:

Algoritmo *Fuzzy C-Means (FCM)*:	
ENTRADA:	Conjunto de datos de entrada X y número de grupos $K > 2$
SALIDA:	k grupos y sus centros.
1:	Seleccionar un peso exponencial m $(1 < m < \infty)$.
2:	Elegir una matriz de partición inicial $U^{(0)}$ y un criterio de paro ϵ.
3:	Calcular los centros de los grupos difusos $\left\{\mathbf{v}_i^{(l)} \mid i = 1, 2, \ldots, c\right\}$ usando $U^{(l)}$ y ecuación (3.3).
4:	Calcular la nueva matriz de partición $U^{(l+1)}$ usando $\left\{\mathbf{v}_i^{(l)} \mid i = 1, 2, \ldots, c\right\}$ y ecuación (3.4).
5:	Calcular $\Delta = \left\|U^{(l+1)} - U^{(l)}\right\| = \text{máx}_{i,j} \left\|u_{ij}^{l+1} - u_{ij}^{(l)}\right\|$.
6:	IF $\Delta > \epsilon$, entonces $l = l + 1$ e ir al Paso 2.
7:	ELSE $\Delta \leq \epsilon$, entonces parar.

El procedimiento iterativo descrito minimiza la función objetivo en la ecuación (3.2) y converge a un mínimo local.

3.2.2. Algoritmo FCM-SVM^2

Asumiendo que tenemos un conjunto deseado de pares de datos de entrada-salida.

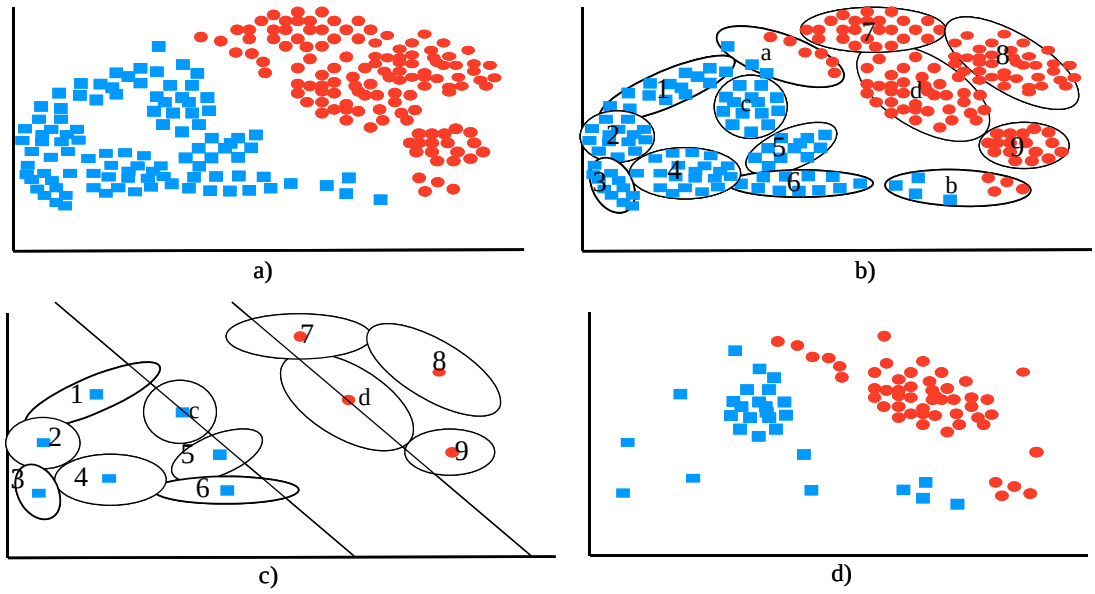

Figura 3.2: Fases del algoritmo FCM-SVM^2

$$X = \{x_1, \ldots, x_n\}, \quad Y = \{y_1, \ldots, y_n\}, \quad y_i = \pm 1, \quad x_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p \qquad (3.5)$$

La tarea consiste en encontrar un conjunto de vectores soporte a partir de los datos de entrada-salida, (ver ecuación 3.5) que maximice el espacio entre clases. Para esto es indispensable:

- Eliminar el subconjunto de datos de entrada más alejado del plano de decisión, evitando eliminar datos que posiblemente sean vectores soporte y,

- Encontrar los vectores soporte a partir del conjunto de datos reducido.

El enfoque consiste de los siguientes pasos:

1) Agrupamiento difuso

De acuerdo a la ecuación (3.5) se asume que se tiene un conjunto de n datos de entrada, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, donde cada dato $\mathbf{x}_i$ puede ser considerado como un punto representado por un vector de dimensión d como se muestra a continuación

$$\mathbf{x}_i = \langle w_{i1}, w_{i2}, \ldots, w_{id} \rangle$$

donde $1 \leq i \leq n$. Aquí w_{ij} denota el grado de importancia del término t_j en $\mathbf{x}_i$, donde $0 < w_{ij} < 1$ y $1 \leq j \leq d$. Además, asumiendo que queremos particionar los datos dentro de k grupos de datos, donde $k \gg 2$ (nótese que el número de grupos a dividir el conjunto original de datos debe ser estrictamente $\gg 2$). La razón de que k sea estrictamente mayor que 2 radica en que deseamos reducir el número de datos de entrada original, eliminando aquellos grupos más alejados del hiperplano de decisión. Si $k = 2$ estaríamos dividiendo el conjunto de datos en el número de clases existentes y no podríamos eliminar algún grupo. Por otro lado, si elegimos un k muy pequeño al eliminar algún conjunto en la segunda etapa, podríamos eliminar algún grupo de datos con datos importantes.

El algoritmo de agrupamiento difuso *C-Means* obtiene k centros y el grado de membresía de cada dato perteneciente a cada grupo de datos, minimizando la función objetivo (3.2) donde u_{ij}^m denota el grado de pertenencia del dato $\mathbf{x}_j$ con respecto al grupo A_k, $\mathbf{v}_i$ denota el centro del grupo A_k y $\|\mathbf{x}_j, \mathbf{v}_i\|$ denota la distancia del dato $\mathbf{x}_j$ al centro $\mathbf{v}_i$. Además, $\mathbf{x}_j$ y $\mathbf{v}_i$ son representados como vectores de dimensión d. Por otro lado, m establece el grado difusividad del algoritmo de agrupamiento. Mientras, más grande sea el valor de m, más grande será el grado de difusividad, i.e., la probabilidad de que un dato pertenezca a múltiples grupos al mismo tiempo es muy grande. Debe notarse que la sumatoria de los grados de pertenencia de cada dato $\mathbf{x}_j$ perteneciente a cada grupo debe ser igual a uno, i.e., $\sum_{i=1}^{k} u_{ij} = 1$ donde $1 \leq i \leq n$. El grado de membresía del dato $\mathbf{x}_j$, que pertenece al grupo A_k, es calculado como en la ecuación (3.4) y el centro v_i del grupo A_k es calculado como en la ecuación (3.3). El algoritmo completo de agrupamiento difuso *C-Means* es desarrollado utilizando los pasos 1 al 7 desarrollados en la sección anterior.

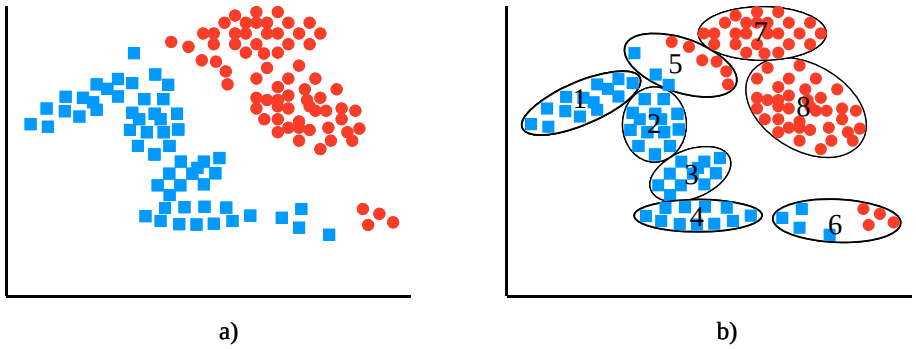

Figura 3.3: Agrupamiento con Fuzzy C-means

2) Clasificación de centros de grupos usando SVM

Dado un conjunto de patrones de entrenamiento (X, Y) donde $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_N\}$ y $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_j, \ldots, \mathbf{y}_N\}$ donde $\mathbf{y}_j \in \{-1, 1\}$, y $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jd})^T \in \mathbb{R}^d$ y cada medida x_{ji} se dice que es una característica. El proceso de agrupamiento difuso se basa en encontrar k particiones de X, $C = \{C_1, \ldots, C_k\}$ $(k < N)$, tal que $a)$ $\cup_{i=1}^{k} C_i = X$ y $b)$ $C_i \neq \varnothing$, $i = 1, \ldots, k$, de donde tenemos que

$$C_i = \{(\mathbf{x}_{i1}, \mathbf{y}_{i1}), (\mathbf{x}_{i2}, \mathbf{y}_{i2}), , (\mathbf{x}_{ik}, \mathbf{y}_{ik})\}, \ i = 1, \ldots, k \tag{3.6}$$

Es decir, independientemente de que cada grupo obtenido contenga elementos con distinto grado de pertenencia o función de membresía a un grupo, cada uno de los elementos posee la pertenencia original a una clase dada, como lo muestra la ecuación (3.6). Los elementos de los grupos obtenidos pueden tener una pertenencia de clase uniforme (como se aprecia en la Figura 3.3(b), en donde los elementos de los grupos 1,2,3,4,7 y 8 tienen un solo tipo de pertenencia). Este tipo de grupos se podría definir como

$$C_u = \{\mathbf{x}_{ij} \mid \mathbf{y}_{ij} = -1 \ \wedge \mathbf{y}_{ij} = 1\}, \ i = 1, \ldots, k$$

o en el caso contrario, una pertenencia de clase mezclada, en donde cada grupo posee elementos de una u otra clase (ver Figura 3.3(b) en donde los grupos 5 y 6 contienen

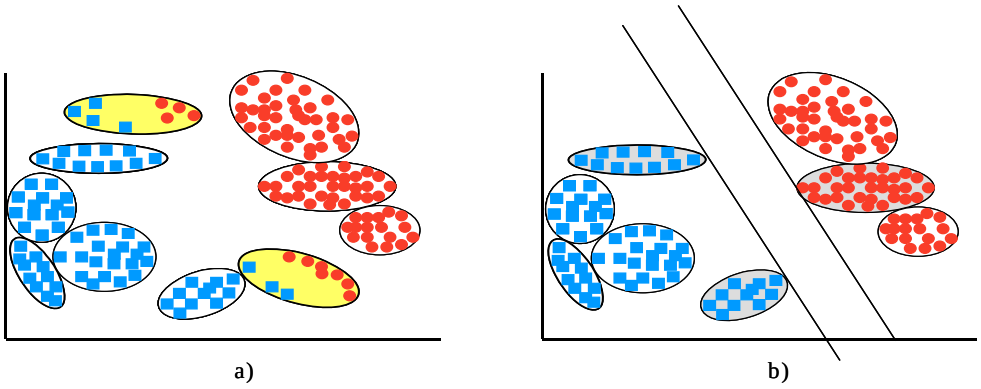

Figura 3.4: Primera etapa de SVM sobre los centros de clusters

elementos con pertenencia de clase mezclada). Este conjunto de datos es definido como

$$C_m = \{\mathbf{x}_{ij} \mid \mathbf{y}_{ij} = -1 \ \vee \mathbf{y}_{ij} = 1\}, \ i = 1, \ldots, k$$

Una vez obtenidos los grupos, se separan aquellos en los que los elementos tienen pertenencia de clase mixta (ver Figura 3.4 *a*)) para una posterior evaluación, debido a que estos contienen elementos con mayor probabilidad de ser vectores soporte. Cada uno de los grupos restantes posee un centro de grupo y una etiqueta de pertenencia de clase uniforme (ver Figura 3.4 *b*)). Utilizando estos centros de grupos y las etiquetas de pertenencia empleamos SVM para encontrar un hiperplano de separación definido mediante un conjunto de vectores soporte, los vectores soporte encontrados son separados del resto como lo muestra la Figura 3.5 *b*).

3) Desagrupamiento

El subconjunto de datos obtenido a partir del conjunto original dado en la ecuación (3.5) está formado por

$$\begin{array}{lll} \cup_{i=1}^{l} C_m, \ l & \leq & k \ \text{ y} \\ \cup_{i=1}^{p} C_u, & \mid & C_u \in svc, \ p \leq k \end{array}$$

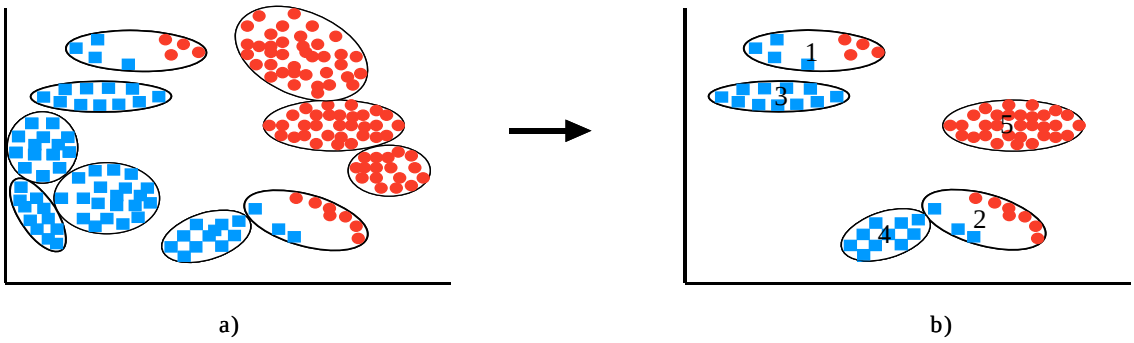

Figura 3.5: Eliminación de grupos irrelevantes

donde *svc* se define como el conjunto de grupos con elementos uniformes que son vectores soporte. C_m y C_u se definen como los grupos con elementos mixtos y uniformes, respectivamente. En la Figura 3.5 *b*), el conjunto de grupos con elementos uniformes y que además son vectores soporte (*svc*) estaría dado por los grupos del 3 al 5. Mientras que conjunto de grupos con elementos de pertenencia mixta estaría dado por los grupos 1 y 2. Hasta este punto, el conjunto de datos original ha sido reducido obteniendo aquellos grupos cercanos al hiperplano de separación óptima y eliminando aquellos que se encuentran alejados de éste. Sin embargo, el subconjunto obtenido está formado por grupos y es necesario desagruparlos y obtener los elementos de éstos.

El subconjunto de patrones obtenido (X', Y'), después de haber desagrupado los grupos obtenidos en el *paso 3,* cumple con las características:

$$\begin{aligned} &1) \quad X' \subset X \\ &2) \quad X' \cap X = X' \\ &3) \quad X' \cup X = X \end{aligned} \tag{3.7}$$

donde estas características se cumplen tanto para X como para Y, i.e., el conjunto de datos obtenido es un conjunto de datos reducido del conjunto de datos original.

4) Clasificación de Datos Reducidos Usando SVM

En este paso, aplicamos SVM sobre el conjunto de datos reducido en los pasos 1 al 3. Ya que el conjunto de datos original es reducido significativamente en los pasos anteriores, eliminando los centros de grupos más alejados del hiperplano de separación, el tiempo de entrenamiento en esta etapa es mucho menor comparado con el tiempo de entrenamiento del conjunto original de datos.

Algoritmo FCM-SVM^2	
ENTRADA:	Conjunto de datos de entrada X y número de grupos $K > 2$
SALIDA:	Vectores Soporte (VS)
1:	Agrupar el conjunto de datos de entrada dentro de K grupos, con $K = (C_u \cup C_m)$.
2:	Separar C_m para evaluarlos en fase final
3:	Aplicar SVM sobre los centros de C_m
4:	Extraer centros de grupos que son vectores soporte
5:	Obtener un subconjunto de datos al desagrupar C_m y $(\cup_{i=1}^{p} C_u, \ \mid C_u \in svc, \ p \leq k)$
6:	Aplicar SVM sobre el conjunto de datos reducido X_r

Ejemplo del algoritmo FCM-SVM^2

Con el objetivo de ilustrar el funcionamiento del algoritmo FCM-SVM^2 y clarificar la idea básica, se considera el caso más simple de clasificación. El conjunto de datos que se usa es muy sencillo con dos características asociadas a cada dato, de tal forma que el conjunto de datos puede ser graficado.

Ejemplo 3.1 *El conjunto de datos es generado aleatoriamente, el conjunto de datos tiene 40,000 datos sin determinar algún tipo de rango o radio de generación aleatoria. El conjunto de datos generado es de dos dimensiones y los datos se etiquetaron basados*

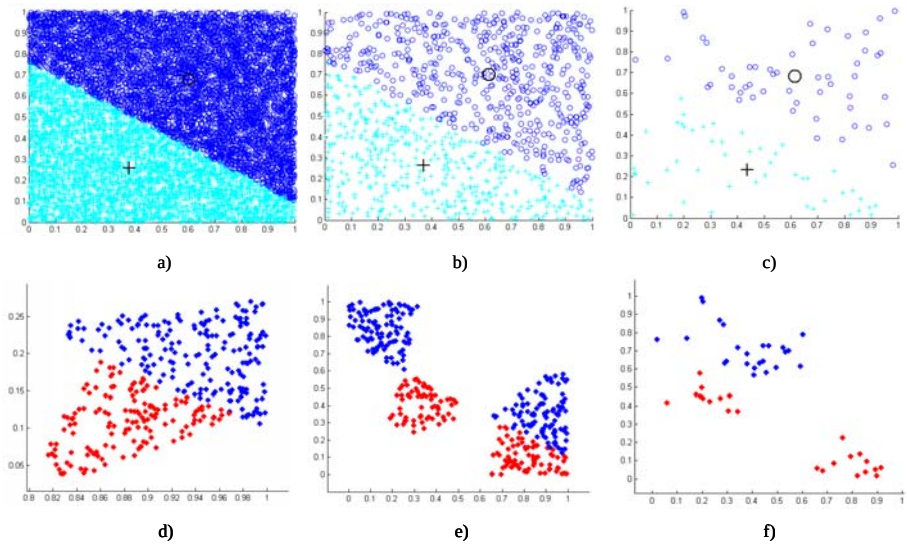

Figura 3.6: Conjuntos de datos de entrada originales a), b) y c) y conjuntos de datos reducidos d), e) y f) con el algoritmo FCM-SVM^2.

en el valor del vector X de cada registro, de tal forma que el registro r_i es etiquetado como positivo si la cota de decisión entre categorías dada por $wx + b$ es mayor que th y negativo si $wx + b$ si es menor que th, donde th es el valor frontera predefinido, w es el vector de pesos asociado a los datos de entrada y b el bias, de esta forma el conjunto de datos es linealmente separable.

En las Figuras 3.6 $a), b)$ y $c)$ se pueden apreciar conjuntos de datos de entrada de diez mil, mil y cien registros respectivamente. Es claro que el ejemplo es muy sencillo y es realizado únicamente para clarificar la idea del algoritmo FCM-SVM^2.

Después de realizar los tres primeros pasos del algoritmo propuesto (agrupamiento, SVM1 y reducción de datos), los conjuntos de datos de entrada son reducidos a 337, 322 y 47 datos respectivamente, ver $d)$, $e)$ y $f)$ en Figura 3.6.

Para probar los resultados obtenidos, generamos un conjunto de 56000 datos. En la Tabla 3.1 se muestra la comparación del desempeño de SVM clásicas y FCM-SVM^2, SMO

Tabla 3.1: Resultados de comparación del algoritmo FCM-SVM^2 con el algoritmo SMO y LibSVM.

SVM clásicas				FCM-SVM^2				SMO		Libsvm	
#	V_S	t	Acc	V_S	t	Acc	k	t	Acc	t	Acc
500	5	96.17	99.8	4	3.359	99.12	10	4.7	99.4	0.17	99.4
1000	5	1220	99.9	4	26.657	99.72	20	11.57	99.8	0.53	99.8
10000	5	24870	100	3	114.78	99.85	50	919.7	100	14	100
20000	-	-	-	3	235.341	100	100	3702	100	56	100
30000	-	-	-	3	536.593	100	150	7969	100	129	100
40000	-	-	-	3	985.735	100	200	17400	100	240	100

número de datos de entrada, V_S vectores soporte, t tiempo de entrenamiento
Acc precisión de clasificación k número de grupos.

y LIBSVM. En este ejemplo se utilizaron SVM clásicas con kernel lineal. Los resultados de comparación de los diferentes desempeños del algoritmo FCM-SVM^2 son mostrados en la Tabla 3.1. Donde "#" representa el número de datos, "V_s" es el número de vectores soporte, "t" es el tiempo total de entrenamiento en segundos, "k" es el número de grupos, "Acc" es la precisión. se puede ver que la reducción de datos es llevada a cabo al utilizar, por vez primera SVM sobre los centros de los grupos y eliminar los más alejados del hiperplano de decisión. Ya que este procedimiento es aplicado sobre los centros de los grupos, esto provoca que algunos vectores soporte sean eliminados. A pesar de esto, el algoritmo propuesto obtiene mejores desempeños y el tiempo de entrenamiento es bastante menor comparado con el tiempo de entrenamiento de las SVM clásicas. El desempeño del algoritmo FCM-SVM^2 es muy competitivo, como se puede apreciar en la Tabla 3.1, en donde en las simulaciones muestran los resultados de tiempo de entrenamiento y precisión comparados con otras implementaciones de recientes SVM.

Las simulaciones con pequeños conjuntos de datos muestran que el tiempo de entrenamiento del enfoque propuesto es mayor que SMO y LIBSVM, sin embargo, cuando el conjunto de datos es muy grande el tiempo de entrenamiento es menor que éstas implementaciones, esto es debido a que muchos datos irrelevantes son eliminados en las primeras dos etapas del algoritmo. Una desventaja notable del algoritmo es que la dimensionalidad de los datos de entrada debe ser menor a 20, las simulaciones realizadas con conjuntos de datos con una dimensión mayor a 20 muestran que el desempeño del algoritmo es afectado considerablemente.

3.3. SVM en dos etapas basado en MEB (MEB-SVM^2)

Los resultados obtenidos en la sección anterior muestran que es posible disminuir el tiempo de entrenamiento, eliminando los puntos más alejados del hiperplano de clasificación y empleando SVM en una primera fase. Es claro, que al eliminar un conjunto de datos, la precisión de clasificación se ve afectada. Sin embargo, ya que el conjunto de datos eliminado está compuesto en su mayoría de datos irrelevantes, la precisión de clasificación es muy similar a la obtenida utilizando el conjunto de datos entero. Sin embargo, a pesar que el algoritmo empleado en la sección anterior disminuye el tiempo de entrenamiento de las SVM de forma importante, su desempeño es bueno únicamente sobre conjuntos de datos de tamaño mediano, i.e., el algoritmo propuesto tiene ciertas dificultades para trabajar con conjuntos de datos muy grandes y alta dimensionalidad. Con el propósito de enfrentar estas desventajas, se propone un algoritmo que trabaja con grandes conjuntos de datos y alta dimensionalidad.

La principal ventaja de esta técnica radica en el hecho de que el conjunto de datos de entrada es reducido enormemente al emplear la fase de *Seccionamiento-SVM-Reducción,* llevándonos a obtener un subconjunto de datos muy pequeño del conjunto de datos entero. Este subconjunto de datos reducido cuenta con la característica de ser el subcon-

junto de datos más representativo del conjunto de datos original, ya que éste es obtenido a partir de los vectores soporte obtenidos al aplicar SVM sobre los centros de grupos o esferas. Es por ello, que a pesar de ser empleado una sola vez sobre el conjunto de datos de entrada, los tiempos de entrenamiento son reducidos significativamente. Aunado a esto, se evita emplear tiempo en agrupar el conjunto de datos dentro de un número de grupos o secciones óptimo y solo se divide el espacio de datos de entrada en un número relativamente grande de grupos, refinando los subconjuntos de datos, obteniendo una mínima parte, pero muy representativa del conjunto original. En otras palabras, el conjunto de datos obtenido al final del proceso está conformado por elementos con mayor probabilidad de ser vectores soporte.

Una segunda ventaja de la técnica radica en el hecho de que el algoritmo no es recursivo; éste es llevado a cabo una sola vez sobre el conjunto de datos de entrada y en este proceso se eliminan aquellos subconjuntos de datos más alejados del hiperplano de separación y se obtienen los más cercanos para después refinar el hiperplano utilizando una vez más SVM. Emplear un algoritmo recursivo, en este caso, provocaría la obtención de un conjunto reducido de datos en la primera fase. Sin embargo, en la segunda fase sería altamente probable que se eliminaran vectores representativos del conjunto de datos de entrada en afán de refinar el hiperplano de separación. Esto es primordial, ya que el proceso es llevado a cabo sin sacrificar precisión al clasificar y en un tiempo de entrenamiento muy pequeño.

Los resultados experimentales sobre conjuntos de datos artificiales y reales prueban el poder del algoritmo propuesto. Los experimentos realizados con SVM-FCM son efectuados con conjuntos de datos de gran tamaño y baja dimensionalidad, mientras que los experimentos realizados con SVM-MEB son realizados con conjuntos de datos de gran tamaño y alta dimensionalidad. Los resultados obtenidos muestran la efectividad y aplicabilidad de los algoritmos propuestos.

3.3.1. Esferas de cerradura mínima (MEB)

Los conceptos básicos del algoritmo propuesto utilizando esferas de cerradura mínima (MEB) [45] son descritos a continuación.

Definición 3.1 *Una esfera con centro c y radio r es denotada como $B(c, r)$.*

Definición 3.2 *Dado un conjunto de puntos $S = \{x_1, \ldots x_m\}$ con $x_i \in \mathbb{R}^d$, la MEB de S es la esfera más pequeña que contiene todos los puntos en S, y ésta es denotada como $MEB(S)$*

Definición 3.3 *Una aproximación-$(1+\epsilon)$ de $MEB(S)$ se define como una esfera $B(c, (1+\epsilon)r)$, $\epsilon > 0$ con $r \leq r_{MEB(S)}$ y $S \subset B(c, (1+\epsilon)r)$*

Definición 3.4 *Un conjunto de puntos Q es un core-set de S si $MEB(Q) = B(c, r)$ y $S \subset B(c, (1+\epsilon)r)$*

Para problemas de agrupamiento, deberán existir varias esferas en el conjunto de datos S. Por lo tanto, la definición de una aproximación $(1+\epsilon)$ de $MEB(S)$ es modificada como:

Definición 3.5 *En agrupamiento, una aproximación $(1+\epsilon)$ de $MEB_c(S)$ es definida como un conjunto de k esferas B_i $(i = 1 \cdots k)$, $S \subset B_1 \cup B_2 \cup \ldots \cup B_k$, donde k es el número de grupos.*

En otras palabras, dado un $\epsilon > 0$, se dice que un subconjunto Q es una aproximación $(1+\epsilon)$ de S para agrupamiento, si $MEB_c(Q) = \cup_{i=1}^{k} B_i(c_i, (1+\epsilon)r_i)$ y $S \subset \cup_{i=1}^{k} B_i(c_i, (1+\epsilon)r_i)$, *i.e.,* Q es una aproximación-$(1+\epsilon)$ de S con un factor de expansión $(1+\epsilon)$.

Con el objetivo de clarificar el funcionamiento del algoritmo considérese un conjunto finito de elementos $X = \{x_1, x_2, \ldots, x_n\}$ en el espacio Euclidiano $p-$ dimensional $\mathbb{R}^p, x_i \in \mathbb{R}^p$. En este ejemplo supóngase que se emplean tres esferas para particionar el

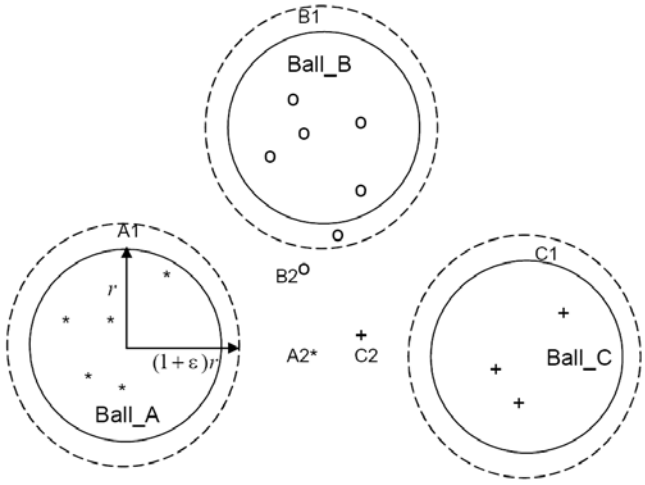

Figura 3.7: Agrupamiento con MEB

conjunto de datos (ver Figura 3.7). Primero, se seleccionan aleatoriamente los centros de las tres esferas a partir del conjunto de datos, de tal forma que éstas cubran todo el rango de los datos. Para ello se emplea la siguiente ecuación

$$r_{ij} = (i - 1 + \text{rand}) \frac{x_{j,\text{máx}} - x_{j,\text{mín}}}{l} \tag{3.8}$$

donde "rand" es un número aleatorio entre $(0, 1)$, $x_{\text{máx}}$ es el máximo de x, $x_{j,\text{máx}} = \text{máx}_i (x_{ij})$, $i = 1 \cdots n$, n es el número de datos, $x_{j,\text{mín}} = \text{mín}_i (x_{ij})$ y l es el número de esferas, donde $l = 3$. Entonces se elige un radio r para las tres esferas, con el objetivo de que el número de iteraciones del algoritmo para obtener la esfera de cerradura mínima no sea muy grande. El radio inicial elegido en este caso es:

$$r = \frac{\|x_{\text{máx}} - x_{\text{mín}}\|}{2l} \tag{3.9}$$

donde $\|x_{\text{máx}} - x_{\text{mín}}\| = \sqrt{\sum_{j=1}^{p} (x_{j,\text{máx}} - x_{j,\text{mín}})^2}$. Entonces se verifica que las tres esferas contengan totalmente a todos los datos. Si esto no ocurre, se alarga el radio en $(1 + \varepsilon)r$. De la Figura 3.7 se puede ver que los puntos A1, B1 y C1 están incluidos en las nuevas esferas (líneas entrecortadas), pero los puntos A2, B2 y C2 aún no son incluidos dentro

del radio de las esferas. De esta forma se alarga un ε hasta que todos los puntos en X se encuentren incluidos dentro de los radios de las esferas, i.e.,

$$X \subset \cup_{i=1}^{l} B\left(c_i, (1+\epsilon) r_i\right)$$

El algoritmo de agrupamiento MEB para seccionamiento de l esferas se puede resumir como sigue:

Algoritmo MEB seccionamiento l esferas	
ENTRADA:	Número de grupos
SALIDA:	Centros de grupos y radios de grupos
1:	*Usar un método de selección aleatoria (3.8) para generar* l *centros de esferas* $C = \{c_1, \ldots, c_l\}$
2:	Seleccionar los radios de las esferas como en la ecuación (3.9)
3:	*Para cada punto* x_i*, calcular la distancia* $\varphi_k(x_i) = \|x_i - c_k\|^2$
4:	Calcular la máxima distancia $\overline{\varphi}(x_i) = \text{máx}_k \left[\varphi_k(x_i)\right]$, $k = 1 \cdots l$, $i = 1 \cdots n$. tal que, $\varphi(q)$ *se encuentre dentro del radio* $B(c_k, (1+\varepsilon) r_k)$ *de algún centro.*
5:	*Terminar el agrupamiento, los grupos son* $B_k\left(c_k, (1+\epsilon) r\right)$, donde $k = 1, 2...n$ es el número de grupos.
6:	*Si existe una* $\overline{\varphi}(x_i) > (1+\varepsilon) r$, $i = 1 \cdots n$ entonces incrementar ε *e ir al paso 4,* $\epsilon = \epsilon + \frac{r}{\Delta}$, donde Δ *es el incremento de cada paso. De otra manera, incluir todos los puntos en las esferas e ir al paso 3.*

3.3.2. Algoritmo MEB-SVM^2

Sea (X, Y) un conjunto de patrones de entrenamiento,

$$X = \{x_1, \cdots, x_n\}, \quad Y = \{y_1, \cdots, y_n\}, \quad y_i = \pm 1, \quad x_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p \quad (3.10)$$

La tarea de entrenamiento de una SVM para clasificación consiste en encontrar el hiperplano óptimo a partir de las entradas X y las salidas Y, el hiperplano óptimo es definido mediante únicamente un pequeño conjunto de vectores soporte, a esta propiedad se le conoce como propiedad de *sparse* de las SVM, los datos que no son vectores soporte no contribuyen de manera alguna al hiperplano óptimo. Luego, es posible eliminar el conjunto de datos de entrada más alejado del hiperplano sin afectar la precisión de clasificación y/o generalización, mientras que el conjunto de datos, que son vectores soporte, se mantiene intacto para un refinamiento en una segunda etapa de SVM.

El algoritmo de clasificación propuesto en este Capítulo (Algoritmo MEB-SVM^2) puede ser resumido en 4 pasos, los cuales se muestran en la Figura 3.8 a) partición del conjunto de datos de entrada, Figura 3.8 b) la primer etapa de clasificación con SVM tomando como datos de entrada los centros grupos, Figura 3.8 c) desagrupamiento, Figura 3.8 d) la segunda etapa de clasificación con SVM. Las siguientes subsecciones dan una explicación detallada de cada paso.

1) Agrupamiento de datos usando esferas de cerradura mínima

Con el objetivo de emplear SVM, es necesario seleccionar datos como un conjunto de datos de entrada inicial a una SVM a partir del conjunto original de datos. En el enfoque expuesto en este Capítulo, se proponen dos alternativas de métodos de selección de datos, en donde las dos técnicas incluyen: partición de datos de entrada empleando MEB y selección aleatoria.

En esta etapa, el algoritmo propuesto secciona el conjunto de datos entero en un conjunto de esferas empleando el algoritmo Esferas de Cerradura Mínima (MEB), después del agrupamiento con MEB, existen l esferas con un radio inicial r y una aproximación $(1+\epsilon)$ de MEB de (S). Los grupos obtenidos mediante agrupamiento pueden ser clasificados en tres tipos y éstos son denotados por Ω^+, Ω^- y Ω_m, donde Ω^+ consiste de grupos con únicamente etiquetas positivas, Ω^- contiene grupos con únicamente datos con etiquetas negativas, el conjunto de centros de grupos en Ω^+ y Ω^- son denotados por C^+ y C^-. Ω_m consiste de grupos con datos con ambas etiquetas, tanto positivas como

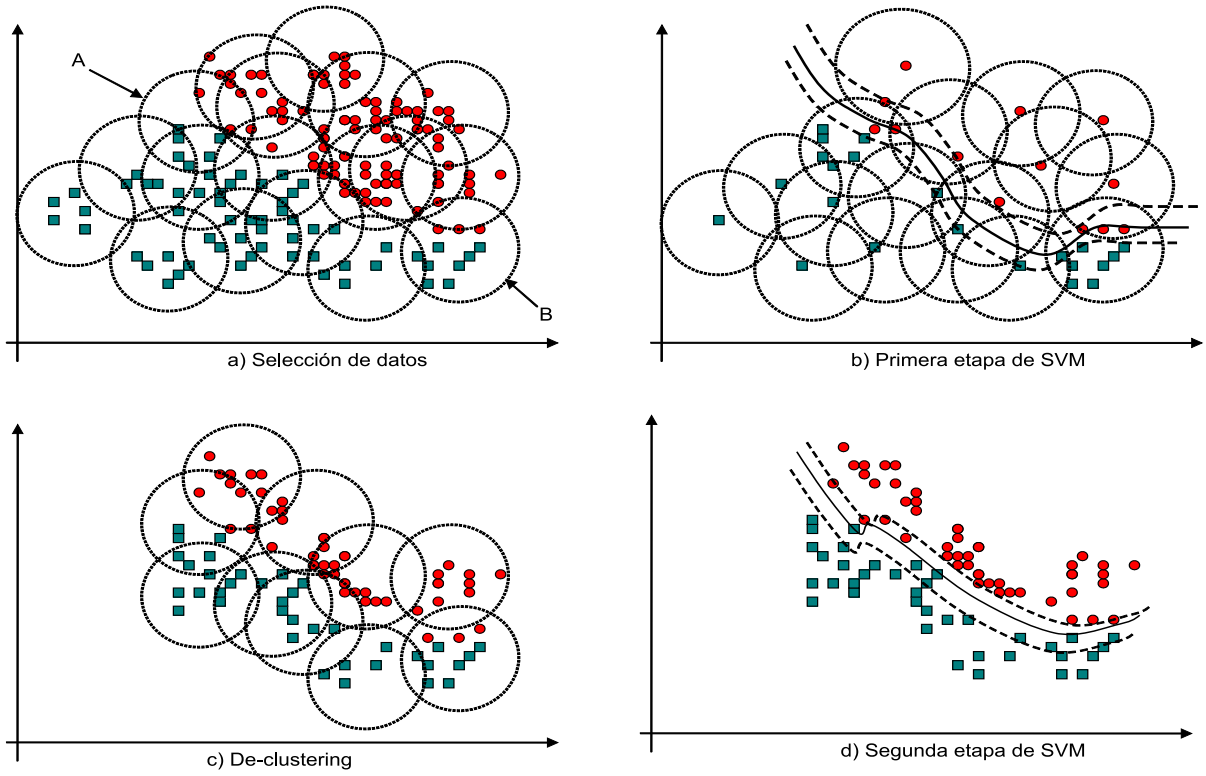

Figura 3.8: Etapas del algoritmo MEB-SVM^2

negativas. i.e.,

$$\begin{aligned}\Omega^+ &= \{\cup\Omega_i \mid y = +1\} \text{ grupos con etiquetas positivas} \\ \Omega^- &= \{\cup\Omega_i \mid y = -1\} \text{ grupos con etiquetas negativas} \\ \Omega_m &= \{\cup\Omega_i \mid y = \pm 1\} \text{ grupos con etiquetas mixtas}\end{aligned} \tag{3.11}$$

En la Figura 3.8 a), los puntos rojos representan etiquetas de datos positivas, los puntos azules representan etiquetas de datos negativas. De acuerdo a la clasificación anterior, los grupos azules pertenecen a Ω^-, los grupos rojos pertenecen a Ω^+, y los grupos 1, 2 pertenecen a Ω_m, ya que existen tanto puntos con etiquetas positivas como negativas en estos dos grupos.

Los grupos con datos de etiquetas mixtas son importantes para la clasificación, ya que éstos poseen una mayor probabilidad de ser vectores soporte. Por lo tanto, no solo se usan los centros de grupos que son vectores soporte, sino también todos los grupos con etiquetas mixtas i.e., los datos seleccionados que serán usados en la primera etapa de clasificación con SVM son la unión de C^+, C^- y los datos en Ω_m.

2) Primera etapa de clasificación con SVM

Se asume que el conjunto original de datos es dividido en l grupos, $l > 2$. Si $l = 2$, entonces los grupos obtenidos en el *Paso 1* son las dos clases existentes, no existiendo reducción alguna de datos. Aquí se considera el caso $l >> 2$.

Nótese que el tamaño de datos original n ha sido reducido en $l + m$ grupos después del *Paso 1*, donde m es el número de datos dentro de los grupos con etiquetas mixtas Ω_m y l es el número de centros. Ahora se toma este conjunto de datos reducido como entrada y se emplea clasificación con SVM con el algoritmo de SMO, con el objetivo de obtener el hiperplano óptimo de decisión y el conjunto de vectores soporte V en el primer paso. En la segunda etapa, el conjunto de datos de entrenamiento de agrupamiento con MEB está dado por

$$C^+ \cup C^- \cup \Omega_m \tag{3.12}$$

y el conjunto de datos de entrenamiento de selección aleatoria

$$C^+ \cup C^- \tag{3.13}$$

Es claro que si los radios de los grupos son muy grandes y $2 \ll l$, entonces $m+l \ll n$, porque ahora el número de datos en los grupos es mucho más grande que el número de grupos. Las Figuras 3.8 b) y 3.8 c) muestra los resultados de la primera etapa de clasificación con SVM.

3) Desagrupamiento

En la primera etapa de clasificación con SVM, se emplean únicamente los datos seleccionados en el *Paso 1*, en consecuencia el hiperplano de decisión obtenido no puede ser muy preciso. Sin embargo, este nos da una referencia de los grupos que pueden ser eliminados. En este paso, se refina la clasificación usando los datos originales dentro de los grupos cercanos al hiperplano, i.e., un proceso de desagrupamiento es necesario para recuperar los datos originales de estos grupos.

Para agrupamiento MEB, se emplean los datos de los grupos, cuyos centros son vectores soporte, i.e.,

$$\cup_{C_i \in V} \{\Omega_i\}$$

donde V es el conjunto de vectores soporte. Sin embargo, no se obtienen los datos de cada grupo en su totalidad, sino únicamente aquellos que se encuentran más cercanos a los vectores soporte, en este caso, en lugar de recuperar los datos de cada esfera con un radio r_k, donde k representa el $k-ésimo$ radio de la esfera, recuperamos únicamente los puntos cercanos a los centros que son vectores soporte con un radio definido por el margen obtenido en la primera etapa de clasificación con SVM. Al mismo tiempo, los grupos con etiquetas mixtas necesitan ser usados una vez más, ya que éstos contienen datos muy cercanos al hiperplano. Por lo tanto, el conjunto de datos de entrenamiento total en la segunda etapa de clasificación con SVM está dado por

$$\cup_{C_i \in V} \{\Omega_i\} \cup \Omega_m \tag{3.14}$$

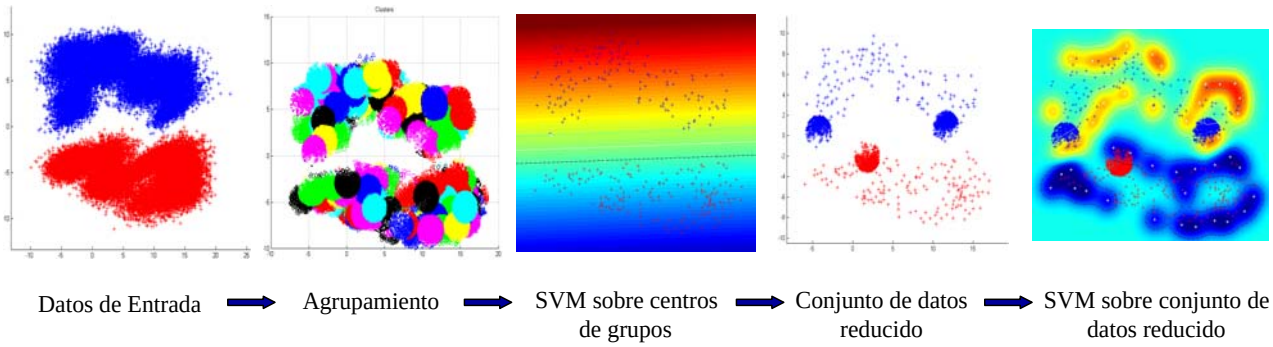

Figura 3.9: Fases del algoritmo MEB-SVM^2.

donde $\forall C_i \in V$ tiene un radio $r_i = M$. Después del proceso de desagrupamiento, los datos originales cercanos al hiperplano pueden ser encontrados. Los resultados de desagrupamiento con MEB son mostrados en la Figura 3.8 c). El proceso de desagrupamiento supera la desventaja de entrenar datos muy alejados del hiperplano óptimo y únicamente se entrenan vectores cercanos al hiperplano.

4) Segunda etapa de clasificación con SVM

Tomando los datos en la ecuación (3.14) como datos de entrada, se emplea una vez más clasificación con SVM, obteniendo un hiperplano de decisión final

$$\sum_{k \in V} \alpha_k y_k K\left(x_k, x_j\right) + b = 0$$

Para el agrupamiento con selección aleatoria, en la primer etapa de clasificación con SVM se obtiene un hiperplano óptimo a partir de los centros seleccionados aleatoriamente $C^+ \cup C^-$ como

$$f(x, \alpha^*, b^*) = \sum_{i \in V} y_i \alpha_i^* \left\langle x_i \cdot x \right\rangle + b^*$$

Los datos de entrenamiento para la segunda etapa de clasificación con SVM contienen tanto los vectores soporte encontrados en la primera etapa de clasificación como aquellos puntos cercanos a los vectores soporte encontrados. Con el objetivo de encontrar los

puntos cercanos al área donde se encuentran los vectores soporte ya encontrados, se usa la siguiente ecuación para calcular el radio que contiene un conjunto de puntos cercanos.

$$B_i\left(c_i, r\right), \quad c_i \in V, \quad \gamma_i = \frac{1}{\|w^*\|_2} = \frac{1}{\left\|\sum_{i \in V} y_i \alpha_i^* x_i\right\|_2}$$

Entonces, los datos de entrenamiento para la segunda etapa de clasificación con SVM está dado por

$$\cup_{C_i \in V} \left\{B_i\left(c_i, r\right)\right\}$$

Nótese que el conjunto de datos original es reducido significativamente al emplear agrupamiento, la primera etapa de SVM y desagrupamiento. Los datos seleccionados para la segunda etapa de clasificación deben ser lo razonablemente grandes desde el punto de vista de optimización, porque los grupos cuyos centros están muy alejados del hiperplano de decisión han sido eliminados en la primera etapa. Por otro lado, la mayoría de los datos cercanos al hiperplano han sido conservados al desagrupar los grupos cercanos a éste. El algoritmo MEB-SVM^2 muestra los pasos para la clasificación de datos.

Algoritmo MEB-SVM^2	
ENTRADA:	Conjunto de datos de entrada X y número de grupos $K > 2$
SALIDA:	Vectores Soporte (VS)
1:	Agrupar el conjunto de datos de entrada dentro de K grupos, con el algoritmo MEB.
2:	Separar C_m para evaluarlos en fase final
3:	Aplicar SVM sobre los centros de C_u
4:	Extraer centros de grupos que son vectores soporte
5:	Obtener un subconjunto de datos al desagrupar C$_m$ y $(\cup_{i=1}^{p} C_u, \ \mid C_u \in svc, \ p \leq k)$ con $r = \frac{1}{\|w^*\|_2}$ para C_u.
6:	Aplicar SVM sobre el conjunto de datos reducido X_r

Ejemplo del algoritmo MEB-SVM^2

Con el objetivo de ilustrar el funcionamiento y clarificar la idea básica del funcionamiento del algoritmo MEB-SVM^2, se considera primero el caso más simple de clasificación y agrupamiento.

Ejemplo 3.2 *Los conjuntos de datos (entrenamiento y prueba) son generados aleatoriamente con una distribución uniforme, pero con cotas de decisión irregulares. El conjunto de datos tiene dos dimensiones, se generaron* $500,000$ *datos de entrenamiento y 100,000 datos de prueba cuyo radio y rango son obtenidos de la misma forma que en [99]. En este ejemplo se compara el desempeño del enfoque propuesto con LIBSVM [9], SMO [67] y Simple SVM [15].*

Para el conjunto de datos del Ejemplo 3.2, Se realizaron cuatro experimentos para mostrar la selección de datos con agrupamiento basado en MEB. Después del agrupamiento, el conjunto de datos original es reducido considerablemente. La Figura 3.10 muestra los resultados. El conjunto de datos de entrada con $500,000$ registros es reducido a 8201, (ver Figura 3.10 a)→d)). El conjunto de datos con $50,000$ datos es reducido a 2947, (ver Figura 3.10 b)→e)). El conjunto con $25,000$ datos es reducido a 375, (ver Figura 3.10 c)→f)).

En ambas etapas de clasificación, se utilizó el mismo kernel RBF y $c = 10$. Aquí SMO es el método propuesto en [67], "Simple SVM" es el método propuesto en [15], MEB-SVM^2 es el enfoque propuesto. Es claro ver que, la precisión de clasificación del enfoque propuesto es un poco menor que los demás enfoques. Sin embargo, el tiempo de entrenamiento es mucho más pequeño comparado con otras implementaciones de SVM.

Para este Ejemplo**,** los tamaños de los datos de entrenamiento son $1,000$, $25,000$, 500,000, y $500,000$. Una vez más se utilizó kernel RBF, y $r_{ck} = r/5$.

Las siguientes notaciones son usadas en las siguientes Tablas.

"#" es el tamaño del conjunto de datos;

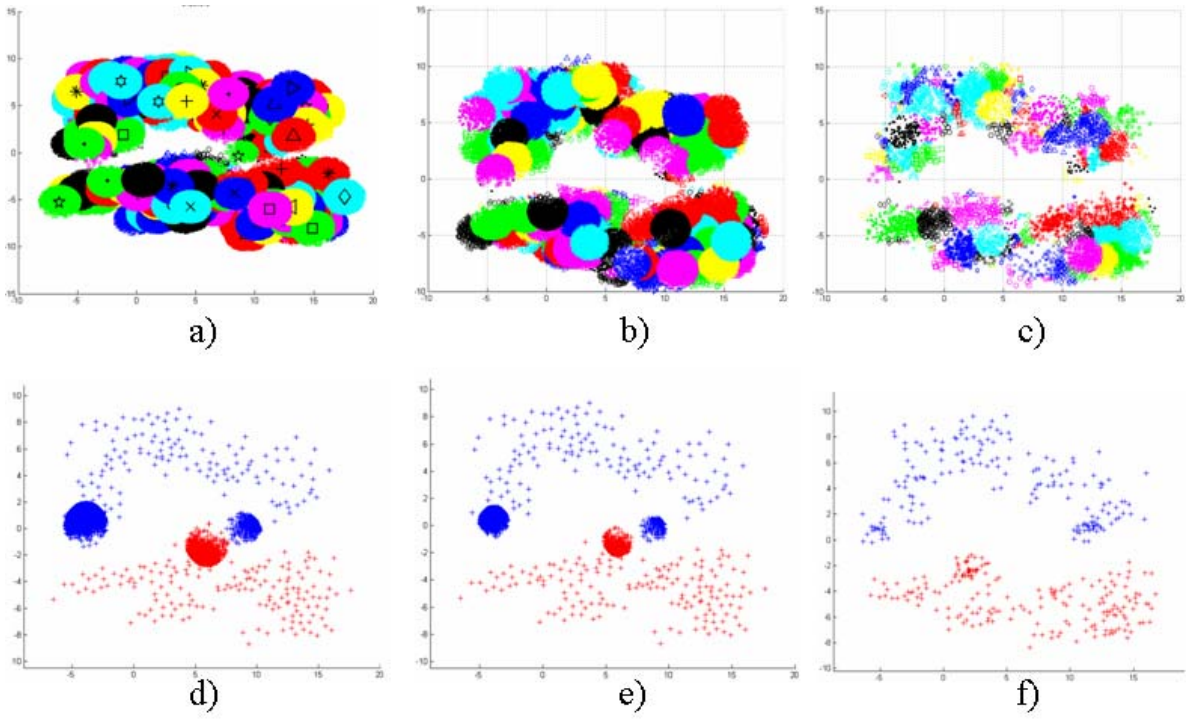

Figura 3.10: Conjuntos de datos originales y conjuntos de datos reducidos mediante el algoritmo propuesto MEB-SVM^2.

Tabla 3.2: Resultados de comparación del algoritmo MEB-SVM^2 con los algoritmos SMO, SSVM y LibSVM.

		MEB-SVM^2		SMO		SSVM		LIBSVM	
Conjunto de datos	#	t	Acc	t	Acc	t	Acc	t	Acc
	1,000	2.6	97.8	14.96	98.1	2.7	98.1	0.3	98
No	25,000	39.1	98.5	6747	98.9	174	98.8	92	98.7
Lineal	50,000	59.3	98.9	25071	99.2	359	99.4	391	99.1
	500,000	275	99.3	–	–	5816	99.5	13670	99.5

número de datos de entrada, t tiempo de entrenamiento, Acc precisión de clasificación

"t" es el tiempo de entrenamiento de todo el proceso de clasificación, incluyendo el tiempo de agrupamiento, la primera etapa de entrenamiento con SVM, el desagrupamiento y la segunda etapa de entrenamiento con SVM para el enfoque propuesto.

"Acc" es la precisión de clasificación;

"l" es el número de grupos empleados en el experimento;

3.4. SVM en dos etapas basado en selección aleatoria (RS-SVM^2)

Los resultados obtenidos con el algoritmo MEB-SVM^2 muestran que el tiempo de entrenamiento es disminuido drásticamente sin afectar la precisión de clasificación. Sin embargo, a pesar que el algoritmo MEB-SVM^2 disminuye el tiempo de entrenamiento de las SVM de forma importante, el algoritmo emplea una gran cantidad de tiempo en seccionar el conjunto de datos, siendo prescindible en algunos conjuntos de datos. Con el propósito de enfrentar estas desventajas, en esta Sección se propone un algoritmo que emplea únicamente un conjunto de datos aleatorio en la primera fase de SVM.

La principal ventaja de esta técnica radica en el hecho de que el tiempo de entrenamiento es reducido enormemente al no emplear la fase de *Seccionamiento* y emplear un conjunto inicial seleccionado aleatoriamente. Al emplear SVM con el conjunto de datos seleccionado aleatoriamente obtenemos un subconjunto de vectores soporte con la característica de ser el subconjunto de datos más representativo del conjunto de datos original, ya que éste es obtenido a partir de los vectores soporte obtenidos al aplicar SVM sobre el conjunto seleccionado aleatoriamente.

Una segunda ventaja de la técnica radica en el hecho de que el algoritmo no recupera completamente los datos de los centros que son vectores soporte, sino únicamente los datos más cercanos con un radio igual a $\frac{1}{\|w^*\|_2}$ y se aquellos datos más alejados del hiperplano de separación. Los resultados experimentales sobre conjuntos de datos artificiales y reales prueban el poder del algoritmo propuesto. Los experimentos realizados con MEB-SVM^2 son efectuados con conjuntos de datos de gran tamaño, sin embargo el tiempo de entrenamiento es afectado considerablemente al seccionar el conjunto de datos, al emplear un algoritmo de selección aleatoria, el tiempo de entrenamiento es reducido de forma considerable. Los resultados obtenidos muestran la efectividad y aplicabilidad de los algoritmos propuestos.

3.4.1. Algoritmo RS-SVM^2

En esta Sección, se propone un algoritmo de agrupamiento de selección aleatoria, con el objetivo de obtener un conjunto de ejemplos aleatorios C a partir de un conjunto de datos X, donde C es el conjunto de centros de grupos. Asumiendo que (X, Y) es un conjunto de patrones de entrenamiento, donde $X = \{x_1, x_2, ..., x_n\}$ es el conjunto de datos de entrada, x_i puede ser representado por un vector de dimensión p, i.e., $x_i = (x_{i1} \dots x_{ip})^T$, $Y = \{y_1, y_2, ..., y_n\}$ representa el conjunto de etiquetas, donde la i-ésima etiqueta $y_i \in \{-1, 1\}$.

El conjunto de datos de entrada X es dividido en dos grupos de acuerdo a su etiqueta Y, i.e., asignando un número de etiquetas positivas como q y un número de etiquetas

negativas como m, se definen las etiquetas de entrada positivas y negativas en forma de arreglos como X^+ y X^- correspondiéndoles las etiquetas Y^+ y Y^- respectivamente, donde $n = q + m$

$$X^+ = [x^+(1), \ldots, x^+(q)]$$
$$X^- = [x^-(1), \ldots, x^-(m)]$$
$$Y^+ = [y^+(1), \ldots, y^+(q)] = [1, \ldots, 1]$$
$$Y^- = [y^+(1), \ldots, y^+(m)] = [-1, \ldots, -1]$$

Por lo tanto, el conjunto de datos de entrada original está dado por la unión de X^+ y X^-, i.e., $X = X^+ \cup X^-$.

Se elige un número de datos de entrada l. Si $l > m$, se selecciona completamente el conjunto X^- como datos de entrada y los datos restantes de forma aleatoria del conjunto X^+. Si $l > q$, se selecciona completamente el conjunto X^+ como datos de entrada y los datos restantes de forma aleatoria del conjunto X^-.

Si $l < q$ y $l < m$, se seleccionan los centros de grupos de forma aleatoria a partir de los subconjuntos X^+ y X^- de manera proporcional e independiente. En esta parte se emplea un método de intercambio o *swapping* para el agrupamiento de selección aleatoria. El primer dato c_1 es elegido a partir de X^+ (o X^-) aleatoriamente y de manera uniforme, entonces es intercambiado con el último dato $x^+(q)$ (o $x^-(m)$), i.e., $Swap(c_1, x^+(q))$ (o $Swap(c_1, x^-(m))$). El segundo centro c_2 es seleccionado de los datos restantes $\{x^+(1), \cdots, x^+(q-1)\}$, entonces es intercambiado con el último dato $x^+(q-1)$ (o $x^-(m-1)$), i.e., $Swap(c_2, x^+(q-1))$ (or $Swap(c_2, x^-(m-1))$), y así sucesivamente hasta que todos los centros para X^+ (o X^-) hayan sido seleccionados.

Es claro que si el número de centros $l = n$, entonces el vector de centros final es una permutación de X. Este método puede ser tenido en cuenta para generar las primeras l entradas en una permutación aleatoria. Por lo tanto es muy conveniente.

El algoritmo de selección aleatoria descrito anteriormente puede ser establecido de la manera siguiente:

Algoritmo obtención de datos de entrada aleatorios	
ENTRADA:	número de datos de entrada l
SALIDA:	l vectores
1.	calcular q y m
1.	*si* $l > m$, seleccionar X^-
2.	seleccionar aleatoriamente $l - m$ datos de X^+
3.	*si* $l > q$, seleccionar X^+
4.	seleccionar aleatoriamente $l - q$ datos de X^-
5.	si $l < q$ y $l < m$
	seleccionar C^+ y C^- a partir de X^+ y X^-

1) Selección de datos

En este algoritmo se emplea el algoritmo de selección aleatoria previamente mencionado para elegir centros de grupos. Por conveniencia, se denotan aquellos datos seleccionados como centros de forma similar a los dos métodos anteriores, De la misma forma, se dividen los datos seleccionados en dos tipos, uno con etiquetas positivas y otro con etiquetas negativas. Esto es,

$$C^+ = \{\cup C_i \mid y = +1\} \text{ datos con etiquetas positivas}$$
$$C^- = \{\cup C_i \mid y = -1\} \text{ datos con etiquetas negativas}$$

entonces se usa C^+ y C^- para la primera etapa de clasificación con SVM.

2) La primera etapa de clasificación con SVM

Se asume que del conjunto original de datos son obtenidos de forma aleatoria l datos, $l >> 2$.

Nótese que el tamaño de datos original n ha sido reducido en l datos después del *Paso 1*. Ahora se toma este conjunto de datos reducido como entrada y se emplea clasificación

con SVM con el algoritmo de SMO, con el objetivo de obtener el hiperplano óptimo de decisión y el conjunto de vectores soporte V en el primer paso. En la segunda etapa, el conjunto de datos de entrenamiento está dado por

$$C^+ \cup C^-$$

donde C^+ y C^- representan el conjunto de datos con etiquetas positiva y negativas respectivamente, obtenidos con el algoritmo de selección aleatoria. En la primer etapa de clasificación con SVM se obtiene una función de decisión a partir de los centros seleccionados aleatoriamente $C^+ \cup C^-$ como

$$f(x, \alpha^*, b^*) = \sum_{i \in V} y_i \alpha_i^* \langle x_i \cdot x \rangle + b^*$$

3) Desagrupamiento

En la primera etapa de clasificación con SVM, se emplean únicamente los datos seleccionados en el *Paso 1*, en consecuencia el hiperplano de decisión obtenido no puede ser muy preciso. Sin embargo, este nos da una referencia de que grupos pueden ser eliminados. En este paso, se refina la clasificación usando los datos originales dentro de los grupos cercanos al hiperplano, i.e., un proceso de reescaneo de datos cercanos al hiperplano es necesario para recuperar datos importantes.

Para agrupamiento MEB, se emplean los datos de los grupos, cuyos centros son vectores soporte, i.e.,

$$\cup_{C_i \in V} \{\Omega_i\}$$

donde V es el conjunto de vectores soporte.

Después del proceso de reescaneo, los datos originales cercanos al hiperplano pueden ser encontrados. El proceso de desagrupamiento supera la desventaja de entrenar datos muy alejados del hiperplano óptimo y únicamente se entrenan vectores cercanos al hiperplano.

4) La segunda etapa de clasificación con SVM

Tomando los datos en la ecuación (3.14) como datos de entrada, se emplea una vez más clasificación con SVM, obteniendo un hiperplano de decisión final mediante:

$$\sum_{k \in V} \alpha_k y_k K\left(x_k, x_j\right) + b = 0$$

Los datos de entrenamiento para la segunda etapa de clasificación con SVM contienen tanto los vectores soporte encontrados en la primera etapa de clasificación como aquellos puntos cercanos a los vectores soporte encontrados. Con el objetivo de encontrar los puntos cercanos al área donde se encuentran los vectores soporte ya encontrados, se usa la siguiente ecuación para calcular el radio que contiene un conjunto de puntos cercanos.

$$B_i\left(c_i, r\right), \quad c_i \in V, \quad \gamma_i = \frac{1}{\left\|w^*\right\|_2} = \frac{1}{\left\|\sum_{i \in V} y_i \alpha_i^* x_i\right\|_2}$$

Entonces, los datos de entrenamiento para la segunda etapa de clasificación con SVM está dado por

$$\cup_{C_i \in V} \left\{B_i\left(c_i, r^*\right)\right\}$$

Nótese que el conjunto de datos original es reducido significativamente al emplear agrupamiento, la primera etapa de SVM y desagrupamiento. Los datos seleccionados para la segunda etapa de clasificación deben ser lo razonablemente grandes desde el punto de vista de optimización, porque los grupos cuyos centros están muy alejados del hiperplano de decisión han sido eliminados en la primera etapa. Por otro lado, la mayoría de los datos cercanos al hiperplano han sido conservados al desagrupar los grupos cercanos. Los pasos del algoritmo RS-SVM^2 son mostrados a continuación:

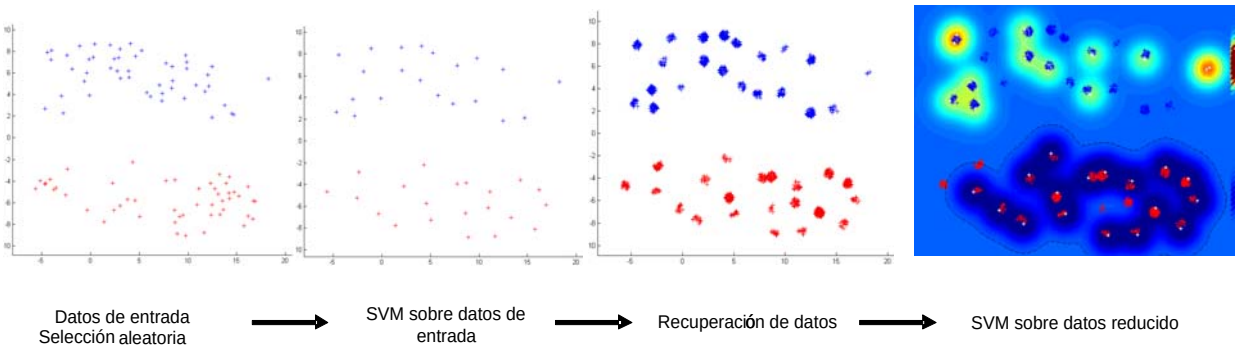

Figura 3.11: Fases del algoritmo RS-SVM^2.

Algoritmo RS-SVM^2	
ENTRADA:	Conjunto de datos de entrada X y número de grupos $K > 2$
SALIDA:	Vectores Soporte (VS)
1:	Seleccionar aleatoriamente K datos
2:	Aplicar SVM sobre los K datos
3:	Obtener puntos cercanos a vectores soporte con radio $r = \frac{1}{\|w^*\|_2}$
4:	Aplicar SVM sobre el conjunto de datos reducido X_r

Ejemplo del algoritmo RS-SVM^2

Los resultados mostrados con el algoritmo anterior muestran que SVM-MEB^2 es un algoritmo muy competitivo con respecto a otras implementaciones actuales de SVM, sin embargo, el tiempo que el algoritmo lleva en seccionar el conjunto de datos de entrada es muy grande. En esta subsección se muestran los resultados obtenidos con el algoritmo SVM-RS^2, en donde se utiliza un conjunto inicial seleccionado aleatoriamente en lugar de seccionar el conjunto de datos entero. Con el objetivo de ilustrar el funcionamiento y clarificar la idea básica del funcionamiento del algoritmo RS-SVM^2, se considera primero el caso más simple de clasificación y agrupamiento.

Tabla 3.3: Comparación de desempeño entre MEB-SVM^2 y RS-SVM^2

		SVM-MEB^2		SVM-RS^2	
Conjunto de datos	#	t	Acc	t	acc
	25,000	39.1	98.5	9.4	98.1
No Lineal	50,000	59.3	98.9	16	98.7
	500,000	275	99.3	98	99.1

datos de entrada, t tiempo de entrenamiento, Acc precisión

Ejemplo 3.3 *Los conjuntos de datos (entrenamiento y prueba) son generados aleatoriamente con una distribución uniforme, pero con cotas de decisión irregulares. El conjunto de datos tiene dos dimensiones, se generaron* $500,000$ *datos de entrenamiento y 100,000 datos de prueba cuyo radio y rango son obtenidos de la misma forma que en [99]. En este ejemplo se compara el desempeño del enfoque propuesto con LIBSVM [9], SMO [67] y Simple SVM [15].*

Para este ejemplo se utilizó kernel RBF, en la Tabla 3.3 se puede ver que la precisión de clasificación del enfoque propuesto es un poco menor que SVM-MEB^2, sin embargo el tiempo de entrenamiento es todavía menor. En la Tabla 3.3 "#" es el tamaño del conjunto de datos; "t" es el tiempo de entrenamiento de todo el proceso de clasificación, que incluye el tiempo de agrupamiento, la primera etapa de entrenamiento con SVM, el desagrupamiento y la segunda etapa de entrenamiento con SVM para el enfoque propuesto y "Acc" es la precisión de clasificación;

3.5. Conclusiones

En este Capítulo se presentaron tres métodos de clasificación con SVM para grandes conjuntos de datos. De acuerdo a la arquitectura de las SVM, únicamente los datos cercanos entre cada frontera son necesarios, resulta claro que si podemos eliminar de una forma eficiente los datos alejados del hiperplano de clasificación antes del entrenamiento, el proceso de entrenamiento puede ser agilizado considerablemente.

Sin embargo, es imposible conocer o tener alguna referencia de los datos cercanos al hiperplano o cuáles están alejados de éste sin haber empleado SVM. La estrategia propuesta, emplea dos fases de clasificación con SVM. Los algoritmos propuestos agrupan el conjunto de datos de entrada o seleccionan un conjunto de datos aleatorio a partir del conjunto de datos de entrada y emplean una primera etapa de SVM con el objetivo de encontrar los vectores soporte de este conjunto de datos. Aún cuando el hiperplano obtenido es únicamente un esbozo del hiperplano real, ya que fue obtenido a partir de un conjunto de datos reducido, éste es de gran utilidad para seleccionar y recuperar todos los datos más importantes del conjunto de datos original. Una vez obtenido el conjunto de datos reducido, se emplea una segunda etapa de clasificación con SVM, esta segunda etapa refina el hiperplano obtenido inicialmente y mejora su capacidad de generalización.

Los algoritmos propuestos reducen considerablemente el tiempo de entrenamiento eliminando datos irrelevantes en el conjunto de datos original y recuperando los más importantes (cercanos al hiperplano de clasificación). Sin embargo, aún cuando los tres emplean la misma estrategia de clasificación, éstos tienen un desempeño muy diferente entre sí. El algoritmo FCM-SVM^2 se desempeña bien en conjuntos de datos grandes pero con dimensiones del espacio de datos de entrada es pequeño, a partir de las simulaciones realizadas se puede inferir que el algoritmo reduce eficazmente el conjunto de datos de entrada eliminando los datos menos representativos en conjuntos de datos con menos de 20 características asociadas a cada registro. Por la naturaleza del algoritmo FCM-SVM^2 se puede inferir que su desempeño podría ser muy bueno en problemas de multiclasificación. Por otro lado, el algoritmo MEB-SVM^2 está diseñado para conjuntos

Tabla 3.4: Comparación entre algoritmos propuestos

	FCM-SVM^2	MEB-SVM^2	RS-SVM^2
AS	FCM	MEB	Aleatorio
TCD	Mediano	Grande	Grande
d	Pequeña	Grande	Grande
CR	G_m y G_{sv}	G_m y G_{sv} con $r = \frac{1}{\|w^*\|_2}$	sv con $r = \frac{1}{\|w^*\|_2}$

AS algoritmo de seccionamiento, TCD Tamaño del conjunto de datos, d dimensión del conjunto, CR Conjunto recuperado, G_m grupos mixtos, G_{sv} grupos que son vectores soporte, r radio.

de datos muy grandes y con alta dimensionalidad, el algoritmo es mucho más rápido que FCM-SVM^2 y reduce eficazmente el conjunto de datos de entrada en conjuntos de datos con distribución uniforme. Finalmente, el algoritmo RS-SVM^2, al igual que MEB-SVM^2, se desempeña eficazmente en conjuntos de datos muy grandes y con alta dimensionalidad ya que no secciona el conjunto de datos de entrada en esferas, sino que selecciona un conjunto de puntos de forma aleatoria, el tiempo de entrenamiento es reducido significativamente.

La principal contribución de los algoritmos propuestos es la prueba de que el tiempo de entrenamiento de las SVM puede ser reducido mediante seccionamiento del conjunto de los datos de entrada y un reescaneo del conjunto de datos original. Esto ha sido realizado, empleando el método de agrupamiento difuso *Fuzzy C-Means*, esferas de cerradura mínima (MEB) y selección aleatoria.

La Tabla 3.4 muestra las diferencias fundamentales entre los algoritmos propuestos como son el algoritmo de seccionamiento empleado en la primera fase para seccionar el

conjunto de datos de entrada, tamaño del conjunto de datos sobre los cuales muestra un mejor desempeño, dimensionalidad del conjunto de datos y los grupos de datos que son recuperados.

Capítulo 4

Resultados experimentales

Con el objetivo de mostrar la validez, precisión y rapidez de clasificación de los algoritmos propuestos, en este Capítulo se muestran los resultados experimentales obtenidos sobre conjuntos de datos artificiales y conjuntos de datos reales. Todos los conjuntos de datos utilizados son regularmente empleados para validar algoritmos de clasificación con SVM. En este Capítulo se muestran y discuten los resultados experimentales obtenidos con los tres algoritmos propuestos FCM-SVM^2, MEB-SVM^2 y RS-SVM^2, descritos en el Capítulo anterior.

Los métodos propuestos son comparados con los métodos de clasificación con SVM más emblemáticos en el estado del arte, como son los algoritmos Simple SVM [96], SMO [67] y LIBSVM [9]. Todos los algoritmos fueron programados como funciones MEX bajo MATLAB y fueron compilados con Microsoft Visual C/C++ 6.0. Ya que las condiciones de trabajo son idénticas para los algoritmos propuestos y los algoritmos comparados, el tiempo de cómputo es considerado como una medida de desempeño.

El objetivo de las SVM es producir un modelo que obtenga las etiquetas de clase de un conjunto de prueba a partir de un conjunto de atributos. Sin embargo, el desempeño de una SVM depende de los parámetros utilizados sobre un conjunto de datos específico. En este Capítulo se describen las técnicas empleadas para obtener un clasificador con una gran habilidad de generalización para cada conjunto de datos, las técnicas empleadas

incluyen procesamiento de datos de entrada, selección de kernels y parámetros, así como estimación de errores de generalización.

El Capítulo está organizado de la siguiente manera: en la Sección 1 se definen los métodos de procesamiento de datos y selección de modelo para obtener un buen clasificador. En la Sección 2 se muestran el desempeño de los tres algoritmos propuestos en el Capítulo anterior sobre conjuntos de datos muy sencillos, el propósito de esta Sección es ilustrar el desarrollo de las fases de entrenamiento de los algoritmos propuestos. En la Sección 3 se muestran los diferentes conjuntos de datos empleados en los experimentos y los resultados obtenidos con cada uno de los algoritmos empleados. Finalmente la Sección 4 muestra las conclusiones obtenidas.

4.1. Procesamiento de datos y selección de modelo

En esta Sección, se definen los diversos métodos empleados para obtener un clasificador con una alta habilidad de generalización. Aunque en esta Sección no se considera la optimización de las características de los datos de entrada, las técnicas empleadas ayudan a mejorar la habilidad de generalización del clasificador, el clasificador con la mejor habilidad de clasificación es llamado clasificador óptimo.

4.1.1. Procesamiento de datos

Cuando las características de los datos de entrada son categóricas, las SVM requieren que cada instancia sea representada como un vector de números reales. Por lo tanto, en los ejemplos que se muestran en esta Sección, cuando existen atributos categóricos, primero convertimos estos a datos numéricos empleando m números para representar un atributo de $m - categorías$. Únicamente uno de los m números es uno, mientras que todos los demás son cero.

Por ejemplo, un conjunto de datos muy popular es *Adult data set,* el conjunto de datos predice si los ingresos de un norteamericano son mayores o menores a 50 mil

Tabla 4.1: Características y valores del conjunto de datos Adult

Característica	valores de la característica
clase de trabajo	9
educación	16
estado civil	14
ocupación	14
raza	5
sexo	2
pais nativo	41

dólares, en base a un conjunto de características continuas y categóricas, sin embargo la mayoría de las características son categóricas como son clase de trabajo, educación, estado civil, ocupación, raza, sexo y país nativo. La Tabla 4.1 muestra las características y los valores asociados a cada una, sin embargo si empleamos un valor entre 1 y 9 para definir la característica clase de trabajo, entre 1 y 16 para educación y asi sucesivamente, la precisión de clasificación que obtenemos es severamente afectada. Si representamos cada base como un vector en donde la presencia de una base es representada por uno y la ausencia por un cero los vectores que representarían raza y sexo para un individuo asiático del sexo masculino serían (0, 0, 0, 1, 0), (1, 0). Es claro que representar un individuo con todas las características categóricas de la Tabla requeriría un vector de 101 datos.

Reescalar los datos de entrada (también llamado normalización de datos) que representan las características es un método muy empleado para obtener clasificadores con una excelente habilidad de generalización. En los conjuntos de datos que empleamos reescalamos cada atributo dentro de un rango de [-1, +1] o [0,1]. Las principales ventajas de reescalar los datos es evitar que los atributos con rangos numéricos muy grandes domi-

nen a aquellos atributos con rangos numéricos pequeños y evitar dificultades numéricas durante los cálculos. De acuerdo a [78], ya que los valores del kernel usualmente dependen de productos punto de los vectores de características (kernel lineal o kernel polinomial), valores con atributos largos pueden causar problemas numéricos. En nuestro caso, los conjuntos de datos de entrenamiento y de prueba son reescalados en un rango [-1, +1] o [0,1] con el objetivo de obtener clasificadores con una mejor habilidad de generalización.

4.1.2. Selección de modelo

La principal desventaja de las SVM además de un largo tiempo de entrenamiento es la selección de parámetros. Para entrenar una SVM es necesario seleccionar previamente un kernel apropiado y un conjunto de valores para el parámetro C. Esto es, para obtener un clasificador óptimo, es necesario determinar el kernel óptimo, así como el valor óptimo de C, al proceso de determinar los parámetros óptimos es llamado selección de modelo.

La selección de modelo usualmente es llevada a cabo estimando la habilidad de generalización para un conjunto de valores de C con un kernel determinado, eligiendo los parámetros para los que el clasificador produce una mejor habilidad de generalización.

Aunque existen diferentes kernels que pueden emplearse para entrenar una SVM, el kernel RBF es uno de los kernels más poderosos para entrenar SVM [5] [14]. El kernel RBF mapea el conjuntos de datos de entrada dentro de un espacio altamente dimensional. Por lo tanto, a diferencia del kernel lineal puede manejar conjuntos de datos en donde la relación entre etiquetas de clase y atributos es no lineal. Además, a diferencia de otros kernels como el polinomial, el número de hiperparámetros es menor, influyendo directamente en la complejidad.

Finalmente, el kernel RBF tiene menos dificultades numéricas. Un punto clave es que el valor del kernel en la $ij - ésima$ posición está dado por $0 < K_{ij} \leq 1$ en contraste con los kernels polinomiales cuyos valores pueden irse al infinito $\left(\gamma x_i^T x_j + r > 1\right)$ o pueden irse a cero $\left(\gamma x_i^T x_j + r < 1\right)$ cuando el grado es grande.

En la mayoría de los ejemplos empleados en este Capítulo se utilizaron previamente

varios kernels con el objetivo de encontrar el clasificador con la mejor habilidad de generalización, en la mayoría de los casos el kernel RBF fue el que mejor desempeño mostró.

4.1.3. Estimación de los errores de generalización

Validación cruzada (*cross validation*) es una técnica ampliamente empleada para estimar el error de generalización de un clasificador. Validación cruzada (también conocida como estimación *leave-one-out*) calcula el error utilizando $n-1$ ejemplos en el conjunto de entrenamiento y probando sobre el ejemplo remanente. El procedimiento es repetido para todos los n subconjuntos de tamaño $n-1$ y la precisión de clasificación para todos los conjuntos de prueba es calculado como la estimación de los n subconjuntos que no son incluidos en los datos de entrenamiento. Sin embargo para un n de gran tamaño esto es computacionalmente caro, requiriendo el entrenamiento de n clasificadores. Sin embargo, en el caso de SVM es posible aplicar este método únicamente a los vectores soporte y la complejidad disminuiría considerablemente, ya que generalmente, los vectores soporte son una pequeña porción del conjunto total de datos. Esto debido a que la precisión de clasificación de las SVM depende de los vectores soporte y no del conjunto total de datos. Sin embargo, aún podría presentar una complejidad muy grande en algunos conjuntos de datos en donde el conjunto de vectores soporte es muy grande, por esa razón usamos *k-fold* validación cruzada.

En *k-fold* validación cruzada, dividimos el conjunto de entrenamiento en k subconjuntos de igual tamaño. Iterativamente un subconjunto es probado empleando el clasificador entrenado sobre los remanentes $k-1$ subconjuntos. Algunos autores [32][9] afirman que el procedimiento de validación cruzada puede prevenir el sobreentrenamiento (*overfitting*). La Figura 4.1 muestra un problema de clasificación binaria, mediante esta Figura ilustramos el problema de sobreentrenamiento. Los círculos y rectángulos rellenos muestran los datos de entrenamiento, mientras que los círculos y rectángulos huecos muestran los datos de prueba. La habilidad de generalización del clasificador de las Figuras 4.1

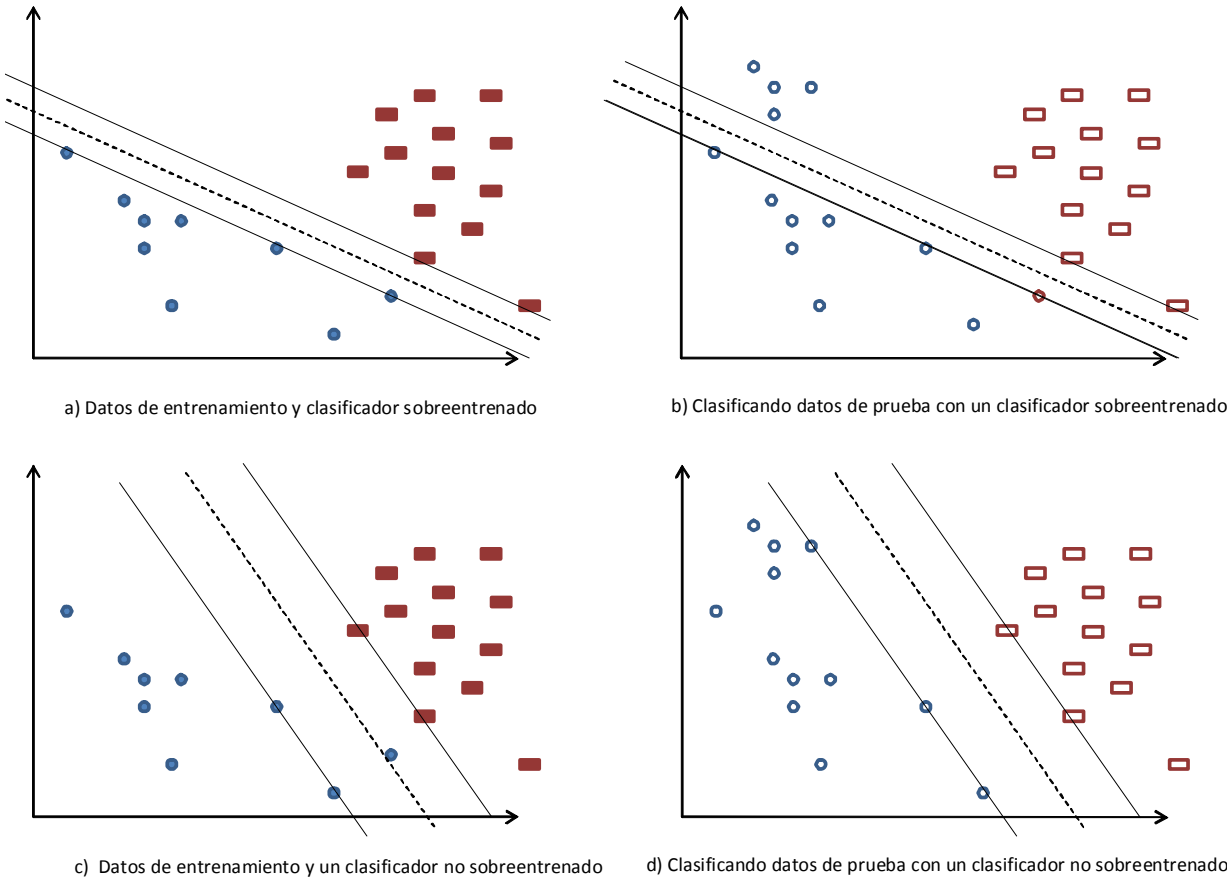

Figura 4.1: Clasificadores sobreentrenados y no sobreentrenados y su habilidad de generalización

a) y 4.1 b) no es muy buena, ya que esta sobreentrena los datos de entrenamiento. Por otro lado, las Figuras 4.1 c) y 4.1 d) muestran un clasificador no sobreentrenado que se desempeña mejor al generalizar.

En SVM el objetivo principal de usar validación cruzada es identificar los mejores parámetros C y γ, de tal forma que el clasificador pueda predecir de forma precisa datos desconocidos (datos de prueba). En todos los conjuntos de datos mostrados en este Capítulo empleamos validación cruzada y búsqueda de malla para escoger los mejores parámetros.

4.2. Experimento en conjuntos medianos y grandes

Debido a las características de los algoritmos propuestos, se emplearon conjuntos de datos de tamaño mediano y gande. Los conjuntos de datos de mediano tamaño son empleados para probar el algoritmo FCM-SVM^2, mientras que los conjuntos de datos de gran tamaño son empleados para probar los algoritmos MEB-SVM^2 y RS-SVM^2.

4.2.1. Conjuntos de datos de tamaño mediano

A continuación se describen los conjuntos de datos de tamaño mediano, número de instancias, número de atributos y características de los conjuntos. En la Tabla 4.2 se muestran los conjuntos de datos de tamaño mediano, la Tabla 4.8 muestra los conjuntos de datos de gran tamaño, características y número de clases.

Fourclass. Es un conjunto de datos artificial con 862 datos en dos dimensiones, cada dato pertenece a una de dos clases. Los datos se traslapan sin mezclarse entre clases.

SVMguide1. El conjunto de datos SVMguide1 contiene 3089 datos de entrenamiento y 4000 datos de prueba, cada dato es representado mediante 4 atributos y pertenece a una de dos clases.

Tabla 4.2: Instancias, atributos y clases de conjuntos de datos de tamaño mediano

Conjunto de datos	Instancias	Atributos	Clases
Fourclass	862	2	2
SVMguide1	3089	4	2
Waveform	5000	21	2
Artificial	40,000	2	2
Breast Cancer	683	10	2

Waveform. El conjunto de datos Waveform contiene 5000 datos que representan ondas, cada ejemplo pertenece a una de tres clases. Las clases son generadas a partir de una combinación de dos o tres ondas base. Cada registro tiene 22 atributos con valores continuos entre 0 y 6 de los cuales todos poseen ruido (con media 0 y varianza 1) en cada atributo.

Wisconsin Breast Cancer. El popular conjunto de datos *Wisconsin Breast Cancer* contiene 10 características 683 ejemplos y ocho clases, los datos de entrenamiento están formados por 376, 70, 31, 17, 41, 40, 31 y 86 ejemplos pertenecientes de la primera a la octava clase respectivamente. Sin embargo en nuestro caso empleamos solo dos clases; la clase 1 contra el resto. El problema es para predecir cuando un conjunto de síntomas tomados de un seno de un paciente presentan un cáncer maligno o benigno.

Artificial1. En este ejemplo, se generaron dos conjuntos de datos en dos dimensiones de forma aleatoria: un conjunto de 40,000 registros para entrenamiento y un conjunto con 5000 como conjunto de prueba para probar los resultados obtenidos. En este ejemplo las cotas de decisión no son irregulares.

Tabla 4.3: Características sin normalizar del conjunto de datos SVM-Guide

x_1	x_2	x_3	x_4	*clase*
23.6567	20.7155	0.1683	98.4939	-1
35.9080	143.4060	0.1315	140.5036	1
0.4395	26.8323	0.3092	44.3635	-1
43.5130	209.3230	-0.1875	136.4990	1
42.1870	286.7140	0.0985	109.7743	1

Las simulaciones fueron realizadas siguiendo los pasos de procesamiento de datos y selección de modelo definidas en la Sección 4.1. Un ejemplo claro que nos muestra como influye el procesamiento de datos en la precisión de clasificación es el siguiente; El conjunto de datos *SVMguide1* contiene 4000 datos, cada dato pertenece a una de dos clases y es representado por 4 características, a continuación se muestran cinco vectores del conjunto *SVMguide* con sus cuatro características y la clase a la que pertenece cada uno:

La Tabla 4.3 muestra únicamente cinco vectores de todo el conjunto de datos, los valores de la primera característica oscilan entre 0 y 244.46, los de la segunda característica entre -7.74 y 546.04, los de la tercera característica entre -0.8016 y 0.7261 y los valores de la cuarta característica oscilan entre 5.64 y 180. Sin embargo, si entrenamos el conjunto de datos sin escalar las características entre 0 y 1 o entre -1 y 1 la precisión de clasificación es severamente afectada. La Tabla 4.4 muestra las precisiones de clasificación obtenidas con diferentes implementaciones de SVM sin procesar el conjunto de datos de entrada.

Sin embargo, si reescalamos cada una de las características del conjunto de datos entre 0 y 1 o entre -1 y 1, la precisión de clasificación es mejorada significativamente, como es posible apreciarlo en la Tabla 4.6, reescalando los datos de entrenamiento estos

Tabla 4.4: Precisión de clasificación sin escalar datos de entrada

Conjunto	SMO	SSVM	LibSVM	FCM-SVM^2
	Acc	Acc	Acc	Acc
SVMguide1	51	**57**	56	51

Acc = precisión de clasificación

Tabla 4.5: Características normalizadas del conjunto de datos SVM-Guide

x_1	x_2	x_3	x_4	*clase*
0.0968	0.0514	0.6349	0.5300	-1
0.1469	0.2729	0.6108	0.7723	1
0.0018	0.0624	0.7271	0.2179	-1
0.1780	0.3920	-0.5980	0.7492	1
0.1726	0.5317	0.5892	0.5951	1

quedarían como en la Tabla 4.5:

En todos los conjuntos de datos es empleado un conjunto de datos de entrenamiento y un conjunto de datos de prueba. El conjunto de datos *SVMguide* contiene 3089 datos de entrenamiento y 4000 de prueba, mientras que los otros conjuntos de datos no tienen datos de prueba, en tales casos seleccionamos el 90 % del conjunto total como datos de entrenamiento y el 10 % como datos de prueba.

La Tabla 4.6 muestra los resultados de comparación en tiempo de entrenamiento y precisión entre el primer enfoque propuesto (FCM-SVM^2) y otras implementaciones de SVM incluyendo SMO, Simple SVM y LIBSVM. En la Tabla 4.6, t representa el tiempo de entrenamiento en segundos, mientras que *Acc* representa la precisión de clasificación

Tabla 4.6: Resultados de simulaciones con conjuntos de datos de tamaño mediano

Conjunto	SMO		SSVM		LibSVM		FCM-SVM^2	
	t	*Acc*	*t*	*Acc*	*t*	*Acc*	*t*	*Acc*
Fourclass	1.14	95.34	23.44	**96.51**	**0.39**	93.0	3.18	95.34
SVMguide1	24.75	95.75	212	**96**	**12.3**	95.75	45	95.2
Waveform	104.35	92	3680	**92.3**	**30.18**	89.8	50.93	91.7
Artificial1	17400	**100**	**182**	**100**	240	**100**	985	**100**
Breast Cancer	0.717	**98.53**	42.59	95.54	**0.313**	98.3	1.52	98.2

Acc precisión de clasificación, t tiempo de entrenamiento.

sobre el conjunto de prueba. La Tabla muestra en letras negritas los mejores valores obtenidos tanto en tiempo de entrenamiento como en precisión de clasificación. Es claro ver en la Tabla que el tiempo de entrenamiento con el algoritmo propuesto es mejor que implementaciones como SMO y Simple SVM, sin embargo, LIBSVM obtiene mejores tiempos de clasificación en todos los casos, excepto en el conjunto de datos *Artificial1*, para el cual, el algoritmo Simple SVM es muy rápido debido al número de características asociadas a cada dato. Por otro lado, la precisión de clasificación es muy similar a las otras implementaciones de SVM. La principal ventaja del algoritmo propuesto radica en que el tiempo de entrenamiento es reducido enormemente sin afectar significativamente la precisión de clasificación al generalizar.

La Tabla 4.7 muestra los resultados obtenidos con el método FCM-SVM^2 sobre el conjunto de datos *SVMguide.* En la Tabla 4.7*,* DE representa el tamaño de los datos de entrenamiento, DP el tamaño de los datos de prueba, k el número de datos empleado inicialmente, t el tiempo de entrenamiento total llevado a cabo por el algoritmo propuesto, Acc1 la precisión inicial obtenida sobre el conjunto de prueba a partir el conjunto de datos utilizado inicialmente como datos de entrenamiento y Acc2 la precisión

de clasificación obtenida sobre el conjunto de prueba, una vez que se redujo el conjunto de datos original y se obtuvo únicamente los datos más importantes de todo el conjunto de datos. El tiempo de entrenamiento esta por arriba de otras implementaciones como SMO y LIBSVM, sin embargo la precisión de clasificación es mejor que las otras implementaciones, aunque la media de tiempo de entrenamiento de 39.9 y desviación estándar de 2.25, mientras que la precisión de clasificación obtenida es de 96.81 % con una desviación estándar de 0.8065.

4.2.2. Conjuntos de datos de gran tamaño

A continuación se describen los conjuntos de datos de tamaño mediano, número de instancias, número de atributos y características de los conjuntos. La Tabla 4.8 muestra los conjuntos de datos de gran tamaño, características y número de clases.

Artificial 2. En este ejemplo, se generaron dos conjuntos de datos en dos dimensiones de forma aleatoria: un conjunto de 109,000 registros para entrenamiento y un conjunto con 50000 como conjunto de prueba para probar los resultados obtenidos. Los datos fueron generados de forma aleatoria especificando rango y radio de generación aleatoria como en [99], lo que provoca que las cotas de decisión sean irregulares.

Artificial 3. Para este ejemplo, los conjuntos de datos (entrenamiento y prueba) son generados aleatoriamente con una distribución uniforme, pero con cotas de decisión irregulares. El conjunto de datos tiene dos dimensiones, se generaron $500,000$ datos de entrenamiento y 100,000 datos de prueba cuyo radio y rango son obtenidos de la misma forma que en [99], además existen ciertas intersecciones entre los datos sin llegar a traslaparse.

IJCNN1. El conjunto de datos IJCNN01 es un conjunto de datos usualmente usado para medir el desempeño de algunos métodos de SVM. Este toma el nombre por la conferencia donde fue propuesto por vez primera. El

Tabla 4.7: Resultados de 20 corridas diferentes con RS-SVM^2

	DE	DP	k	t	$Acc1$	$Acc2$
Corrida						
1	3089	4000	100	39.563	55.5	96.25
2	3089	4000	100	40.017	56.0	97.5
3	3089	4000	100	39.845	56.0	96.75
4	3089	4000	100	41.219	57.0	97.25
5	3089	4000	100	37.970	56.25	95.50
6	3089	4000	100	40.532	56.25	**98.00**
7	3089	4000	100	41.172	56.0	96.50
8	3089	4000	100	38.703	56.25	96.00
9	3089	4000	100	41.484	56.00	97.25
10	3089	4000	100	42.86	55.75	97.5
11	3089	4000	100	36.61	55.75	97.25
12	3089	4000	100	38.70	55.75	97.50
13	3089	4000	100	44.81	55.5	96.75
14	3089	4000	100	43.14	55.75	**95.25**
15	3089	4000	100	38.28	55.50	97.00
16	3089	4000	100	40.51	55.50	96.5
17	3089	4000	100	36.15	56.25	96.50
18	3089	4000	100	39.51	55.0	97.75
19	3089	4000	100	41.79	55.75	95.50
20	3089	4000	100	36.87	55.50	97.75

DE datos de entrada, DP datos de prueba, k número de grupos

t tiempo de entrenamiento, $Acc1$ precisión fase1, $Acc2$ precisión fase2

Tabla 4.8: Instancias, atributos y clases de conjuntos de datos de tamaño grande

Conjunto de datos	Instancias	Atributos	Clases
Artificial 2	109,000	2	2
Artificial 3	500,000	2	2
UCI Adult	30956	123	2
IJCNN1	49990	22	2
Covtype Binary	581,012	54	2
ARN	23605	8	2
Magic	19020	10	2
Checkerboard	1,000,000	2	2
Poker	25010	85	2

conjunto de datos está disponible en [70] y [9]. Este cuenta con 49990 datos de entrenamiento y 91701 datos de prueba, cada dato es representado por 22 atributos.

UCI Adult. Es un conjunto de datos extraído de la base de datos del censo de población de 1994, mediante este conjunto de datos se predicen los ingresos anuales de una persona en Estados Unidos, si estos son mayores a 50,000 Dlls anuales se etiqueta como +1 y si son menores a 50,000 Dlls se etiqueta como -1, los atributos son valores discretos y continuos que representan a edad, clase de trabajo, educación, estado civil, ocupación, raza, sexo, horas de trabajo por semana y país nativo entre otros.

ARN. Un conjunto de datos biológicos disponible en [95]. Este es el conjunto de datos ARN. El conjunto de datos contiene 23605 puntos, cada dato

tiene 8 atributos con valores continuos en 0 y 1. El conjunto de datos contiene 3919 ncRNA y 19686 secuencias negativas.

Covtype-Binary. El conjunto de datos *Forest Cover Type* contiene 522,911 datos, cada dato es representado por 54 características, el conjunto de datos original es empleado para multiclasificación. Sin embargo, el conjunto de datos empleado aquí, es binario, cada dato pertenece a una de dos clases. El conjunto de datos es empleado para predecir el tipo de bosque a partir de únicamente las variables cartográficas. El conjunto de datos original no está escalado y contiene columnas de datos binarios (0 o 1) para representar variables independientes cualitativas. Los atributos son valores continuos y discretos que representan elevación, tipo de tierra, distancia horizontal y vertical, inclinación y aspecto entre otros atributos.

Magic. El conjunto de datos contiene 19020 datos y cada dato es representado por un vector de 10 características, que contiene valores reales, el conjunto de datos no está escalado y fue necesario escalar todos los datos a valores entre [0,1] y [-1, 1]. Ya que el conjunto de datos no tiene un conjunto de prueba, se empleó el 90 % de los datos como entrenamiento y el 10 % restante como prueba.

Poker. El conjunto de datos contiene 25010 datos de entrenamiento y 1,000,000 datos de prueba, el conjunto de datos original posee únicamente 10 características con valores categóricos.

La Tabla 4.9 muestra los resultados en tiempo de entrenamiento y precisión de clasificación obtenidos en las simulaciones, los resultados que se muestran en la Tabla muestran los resultados obtenidos de los enfoques propuestos (SVM-RS^2 y SVM-MEB^2) y otras implementaciones de SVM incluyendo SMO, Simple SVM (SSVM en la Tabla) y LIBSVM. En la Tabla 4.9, t representa el tiempo de entrenamiento en segundos, mientras que *Acc* representa la precisión de clasificación sobre el conjunto de prueba. Los mejores valores obtenidos en tiempo de entrenamiento y precisión de clasificación son mostrados en letras negritas en la Tabla. Algunos conjuntos de datos como *Magic, ARN* y *Covtype*

Tabla 4.9: Resultados de simulaciones con conjuntos de datos de tamaño grande

Conjunto	SMO		SSVM		LibSVM		SVM-RS^2		SVM-MEB^2	
	t	Acc	t	Acc	t	Acc	t	Acc	t	Acc
Artificial 2	12700	100	120	100	351	**100**	**54**	99.9	158	100
Artificial 3	72700	99.9	223.5	99.9	1861	**100**	**72**	**99.9**	275	99.9
UCI Adult	39494	84.3	737	84.7	412.8	**84.7**	**140**	83.7	372	84.2
IJCNN1	27032	**98.5**	20491	98.2	461	**98.5**	**220**	98.2	462.4	98.4
Covtype Binary										
ARN	8309	**93.5**	3081	92.7	409	**93.5**	**274**	92.7	374	93.1
Magic	5314	**88.1**	7385	**88.1**	213	**88.1**	**157**	87.9	294	88.0
Checkerboard	196000*	99.8	22371	99.8	95400	**99.85**	**2300**	99.6	5802	99.7
Poker	42372	90.9	34805	90.1	326	90.83	**372**	90.83	581	90.8

fueron escalados a valores entre [0, 1] y [-1, 1], ya que la precisión de clasificación obtenida con el conjunto de datos original no era aceptable. Una vez escalado el conjunto de datos las precisiones obtenidas mejoraron significativamente en todos los casos. El conjunto de datos Poker originalmente es representado mediante únicamente 10 características cada dato, sin embargo, al entrenarlo la precisión de clasificación obtenida no era aceptable, todas las características fueron tomadas como categóricas y se representó cada instancia como un vector de números reales, empleando m números para representar un atributo de $m - categorías$, en el vector únicamente uno de los m números es uno, mientras que todos los demás son cero.

El tiempo de entrenamiento de los enfoques propuestos es, en general, menor al empleado por otras implementaciones de SVM. Sin embargo, ya que el enfoque propuesto reduce el tiempo de entrenamiento dependiendo del número de vectores soporte, cuando

el conjunto de vectores soporte es muy grande, el conjunto de datos recuperado también lo es, por lo tanto el tiempo de entrenamiento no es reducido significativamente.

Las simulaciones fueron realizadas siguiendo los pasos de procesamiento de datos y selección de modelo definidas en la Sección 4.1. La selección del modelo o parámetros de clasificación influye en gran manera en la en la precisión de clasificación. Como ejemplo, tomamos el conjunto de datos *IJCNN1,* el conjunto contiene 49991 datos de entrenamiento, cada dato pertenece a una de dos clases y es representado por 22 características. La Figura 4.2 muestra la precisión de clasificación empleando el algoritmo SVM-RS^2 y kernel RBF, ya que el desempeño de las SVM varía dependiendo del valor de γ, es necesario encontrar el valor óptimo de γ. Ya que el algoritmo SVM-RS^2 emplea dos fases de SVM es necesario optimizar los parámetros empleados en el algoritmo, resultaría sencillo emplear el mismo valor de γ en el kernel para las dos fases, sin embargo, en base a los resultados obtenidos, se ha comprobado que la precisión de clasificación se ve afectada severamente al emplear el mismo valor de γ en las dos fases, siendo necesario encontrar los parámetros para los que la SVM tiene un desempeño óptimo.

En nuestro caso empleamos una búsqueda de malla. La Figura 4.2 muestra la precisión de clasificación obtenida con diferentes valores de γ, en donde $\gamma = 1/k$, para este caso. En la Figura, las abscisas representan el valor de k_1 con valores entre 1 y 10, mientras que las ordenadas representan el valor de γ_2 con valores también entre 1 y 10. Es claro ver que el valor de k_1 puede tomarse entre 1 y 10 sin afectar significativamente la precisión, sin embargo el valor de k_2 puede variar únicamente entre 1 y 2, cualquier valor de k mayor que 2 reduce la precisión de clasificación.

4.3. Conclusiones

En este Capítulo, se presentaron los resultados experimentales con tres nuevos enfoques de clasificación para trabajar con grandes conjuntos de datos, los métodos propuestos emplean técnicas de agrupamiento para reducir el tamaño de los datos. El principal objetivo de diseñar tales algoritmos es resolver el problema asociado a la clasifi-

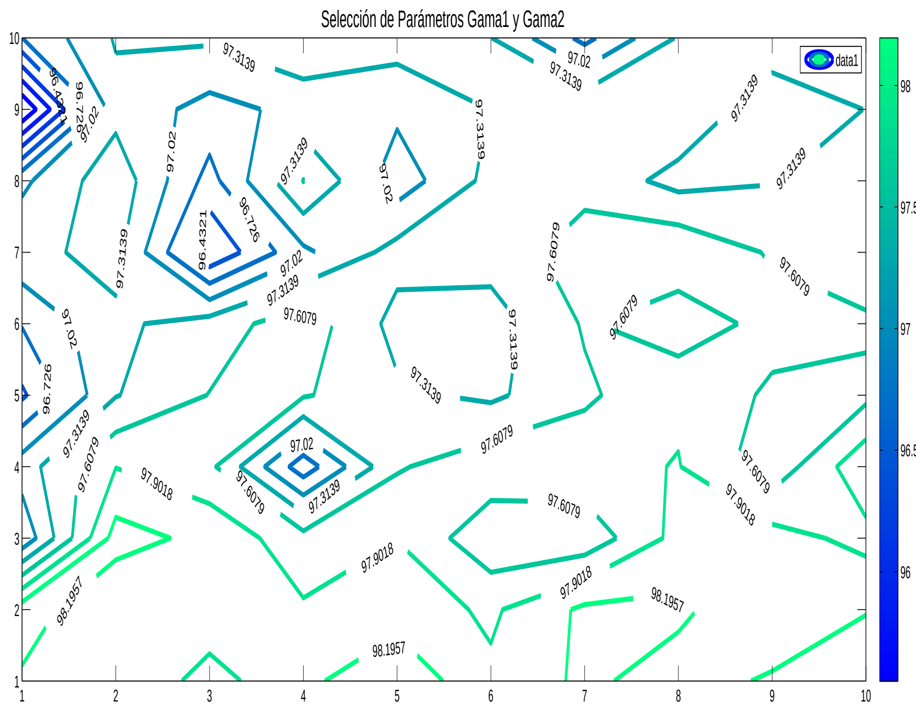

Figura 4.2: Grafica de búsqueda de malla para seleccionar valores de Gama1 y Gama2 del algoritmo RS-SVM^2.

cación con SVM referente a la precisión y al tiempo de entrenamiento. Los resultados experimentales realizados en este Capítulo muestran que los enfoques propuestos poseen una buena precisión de clasificación comparado con otras implementaciones de SVM, mientras que el tiempo de entrenamiento es significativamente más pequeño que otros clasificadores de SVM diseñados para entrenar grandes conjuntos de datos.

Las dos etapas de clasificación con el algoritmo FCM-SVM^2 tienen las siguientes ventajas en comparación con otros clasificadores de SVM:

1. El tiempo de entrenamiento es menor con respecto al tiempo de entrenamiento con SVM clásicas. Es claro que al disminuir el tiempo de entrenamiento se reduce

la precisión de clasificación. Sin embargo, la precisión de clasificación en los casos presentados es similar a la obtenida con otras implementaciones de SVM.

2. Obtener el número de grupo óptimo es computacionalmente muy costoso por lo que el algoritmo propuesto no necesita conocer el número de grupos óptimo, se puede predefinir el número de grupos, siempre y cuando éste sea mucho mayor a 2. Sin embargo, es claro que la relación entre el agrupamiento y los vectores soporte juega un importante rol en la precisión de clasificación. En los experimentos realizados se pudo comprobar que, si bien el número de grupos depende de la distribución del conjunto de datos, en la mayoría de los casos elegir un número de grupos entre el 0.05 % y 0.1 % del conjunto total de datos, da buenos resultados de clasificación.

3. Ya que el agrupamiento no es supervisado, alguna información útil (vectores soporte) del conjunto de datos original puede ser eliminada durante el proceso. Nuevos enfoques de agrupamiento que usen las etiquetas como información pueden llegar a mejorar la precisión. Por ejemplo, la selección aleatoria presentada en esta Sección es realizada sobre dos clases independientes (etiquetas ± 1). Este tipo de método puede ser extendido a métodos generales de agrupamiento semi-supervisado.

El algoritmo MEB-SVM^2 emplea también dos etapas de clasificación con SVM. Sin embargo, este algoritmo emplea esferas de cerradura mínima para seccionar el conjunto de datos de entrada y recuperar los datos cercanos a la frontera de separación entre distintas clases, reduciendo el tiempo de entrenamiento significativamente. El tiempo de entrenamiento del algoritmo MEB-SVM^2 es aún menor con respecto al tiempo de entrenamiento con SMO y LIBSVM. Aunque la precisión de clasificación en los casos presentados es un poco menor que las otras implementaciones, ésta no se ve afectada en gran manera. En los resultados experimentales presentados, es posible apreciar como el tiempo de entrenamiento es reducido significativamente con respecto al algoritmo FCM-SVM^2.

Tabla 4.10: Comparación entre algoritmos propuestos

	FCM-SVM^2	MEB-SVM^2	RS-SVM^2	SMO	SSVM	LibSVM
TCD	Mediano	Grande	Grande	Mediano	Mediano	Grande
DCD	Pequeña	-	-	-	-	-
NP	4	4	4	2	2	2
Acc	MB	MB	MB	Óptimo	Óptimo	Óptimo

TCD Tamaño del conjunto de datos, DCD Dimensión del conjunto de datos, Acc precisión de clasificación, NP número de parámetros a elegir.

Finalmente, el algoritmo RS-SVM^2 tiene una precisión de clasificación muy similar a MEB-SVM^2. Sin embargo el tiempo de entrenamiento es aún menor con respecto a MEB-SVM^2, ya que este método elimina el paso de seccionar el conjunto de datos de entrada y emplea un algoritmo de selección aleatoria para elegir el conjunto de datos inicial.

La Tabla 4.10 muestra las características principales de los algoritmos propuestos, la desventaja más clara de los algoritmos propuestos es elegir un buen conjunto de parámetros para obtener una buena precisión al generalizar.

Capítulo 5

Análisis de desempeño

La gran desventaja de las SVM es el requerimiento de enormes cantidades de tiempo para entrenar conjuntos de datos grandes. De acuerdo a [99], entrenar una SVM con un millón de registros podría tomar años. A pesar de que las últimas implementaciones de SVM como Optimización Mínima Secuencial (SMO), LibSVM y SimpleSVM, entre otras, han reducido enormemente el tiempo de entrenamiento de las SVM clásicas, el tiempo de entrenamiento de las nuevas implementaciones es aún muy grande cuando trabajamos con grandes conjuntos de datos. La Figura 5.1 muestra el conjunto de datos Checkerboard 4x4, el conjunto de datos podría considerarse como un caso sencillo de clasificación debido a que tiene únicamente dos clases en dos dimensiones. Sin embargo, el conjunto de datos tiene muchos puntos de intersección entre clases, como se puede apreciar en la Figura 5.1.

La Fig. 5.2 muestra cuán inescalable puede llegar a ser entrenar una SVM con grandes conjuntos de datos. Considerando el caso más simple de clasificación, un conjunto de datos con dos clases en dos dimensiones, es posible ver cuán costoso puede llegar a ser, entrenar un conjunto de datos con SVM. La Fig. 5.2 muestra los resultados sobre el conjunto de datos Checkerboard empleando las implementaciones de SVM para grandes conjuntos de datos; SMO -a)- y Libsvm -b)- con un conjunto de datos con 1,000,000 registros.

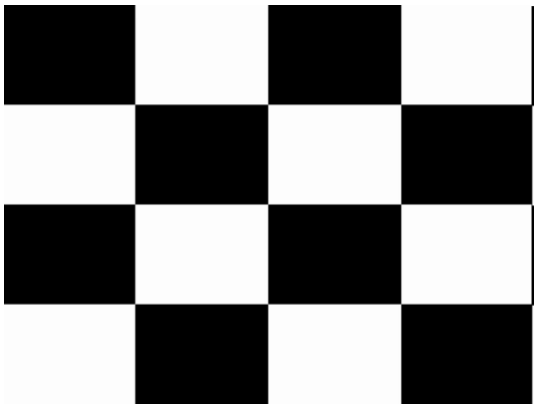

Figura 5.1: Conjunto de datos Checkerboard.

Aunque SMO es una implementación de SVM con un buen desempeño, con grandes conjuntos puede llegar a ocupar demasiado tiempo en entrenar datos. La gráfica muestra los tiempos de entrenamiento, con sólo 100,000 registros el tiempo de entrenamiento con SMO es de 196,780 segundos, mientras que los tiempos de entrenamiento de SimpleSVM y LibSVM son menores, aún con 1 millón de datos, el tiempo de entrenamiento de LibSVM es de casi 100,000 segundos, alrededor de 28 horas. SimpleSVM tiene un menor entrenamiento (23,000 segundos -6.38 horas-), sin embargo, el tiempo de entrenamiento con este algoritmo suele aumentar considerablemente cuando aumenta la complejidad del conjunto de datos.

En esta Sección mostramos la complejidad en tiempo y espacio de los tres algoritmos propuestos. Es claro que sin un método de descomposición, resulta casi imposible para las SVM clásicas obtener el hiperplano óptimo cuando el tamaño de los datos de entrenamiento n es muy grande. Es difícil analizar de forma precisa la complejidad del algoritmo de SVM. Esta operación involucra multiplicación de matrices de tamaño n con una complejidad $O(n^{2,3})$ y $O(n^{2,83})$ en el peor caso. En lo subsiguiente, asumimos que una implementación QP de cada etapa de SVM tiene una complejidad $O(n^3)$ en tiempo y $O(n^2)$ en espacio para n datos de entrada. Por otro lado, la complejidad del algoritmo SMO depende directamente del número de vectores soporte n_{sv}, un resultado

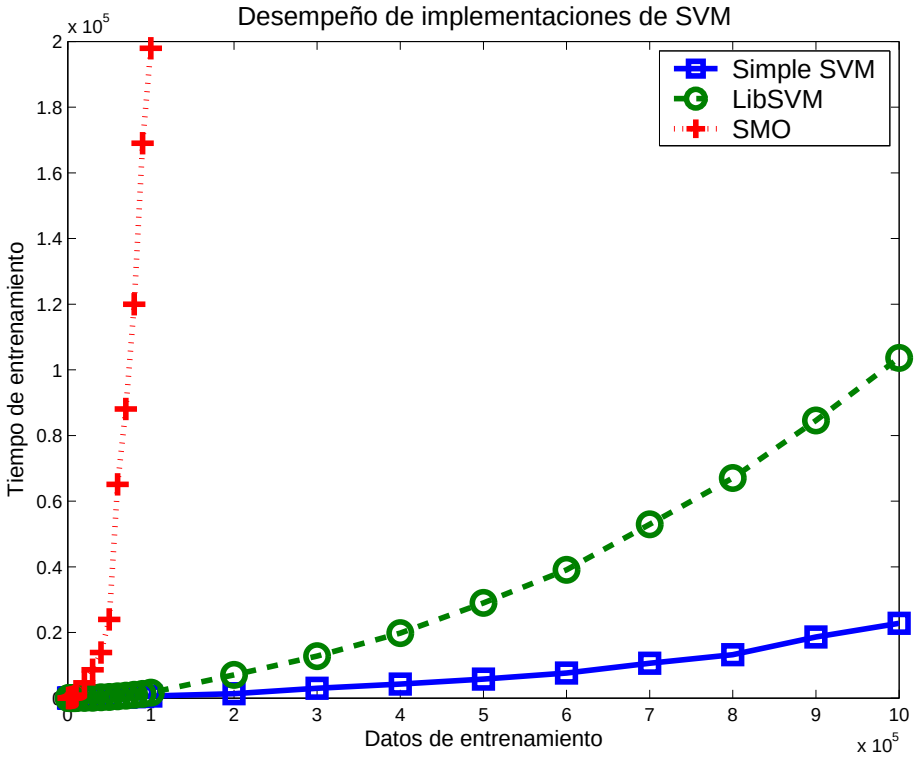

Figura 5.2: Tiempo de entrenamiento de tres implementaciones de SVM sobre el conjunto de datos Checkerboard.

teórico reciente de Steinwart [84] muestra que el número de vectores soporte crece como una función lineal de n. Por lo tanto, para problemas con grandes conjuntos de datos, la complejidad de entrenamiento puede llegar a ser prohibitiva, ya que ésta llega a ser $O(n \cdot n_{sv} + n_{sv}^3)$. Por otro lado, Bottou y Lin [6] sostienen que los algoritmos actuales de SVM se aproximan a $O(n^2)$ cuando el valor de C es pequeño y $O(n^3)$ cuando C es grande, Keerti et al. [41] sostienen que la complejidad está en el orden de $O(L \cdot n)$ donde n es el tamaño del conjunto de datos de entrada y L es el número de candidatos a vectores soporte.

5.1. Algoritmo FCM-SVM^2

Espacio de memoria

En la primera etapa de agrupamiento, el conjunto entero de datos de entrada

$$X = \{x_1, \ldots, x_n\}, \quad Y = \{y_1, \ldots, y_n\}, \quad y_i = \pm 1, \quad x_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p$$

es cargado en la memoria. Los datos son de tipo flotante, por lo tanto el tamaño de cada uno de ellos es de 4 bytes. Si utilizamos SVM clásicas, el tamaño de la memoria para los datos de entrada deberá ser $4\,(n \times p)^2$ por la matriz empleada por el kernel, mientras que el tamaño de los datos de agrupamiento es $4\,(n \times p)$. Es claro que en implementaciones de SVM actuales, no es necesario que la matriz del kernel esté completamente cargada en memoria.

En la primera etapa de clasificación, el tamaño de los datos de entrenamiento es $4\,[(l+m) \times p]^2$, donde l es el número de grupos y m es el número de elementos dentro de los grupos mixtos. En la segunda etapa de clasificación con SVM, el tamaño de los datos de entrenamiento es $4\left[\left(\sum_{i=1}^{l} n_i + m\right) \times p\right]^2$, donde n_i es el número de elementos en los grupos cuyos centros son vectores soporte. El espacio total de agrupamiento con FCM está dado por

$$4\,(n \times p) + 4p^2 \left[\left(\sum_{i=1}^{l} n_i + m\right)^2 + (l+m)^2\right]. \tag{5.1}$$

Cuando n es muy grande (grandes conjuntos de datos), n_i, m y $l \ll n$, el espacio de memoria dado por la ecuación (5.1) del enfoque propuesto es mucho más pequeño que el espacio de memoria empleado por SVM clásicas $4\left(n \times p\right)^2$.

Complejidad algorítmica

Es difícil calcular de forma precisa la complejidad del algoritmo, ya que esta depende en gran parte del conjunto de datos seleccionado inicialmente y del número de vectores soporte del conjunto de datos y no únicamente del tamaño del conjunto de datos, Sin embargo, la complejidad del algoritmo puede ser calculada por partes, de la manera siguiente. La complejidad del agrupamiento usando FCM es $O(ncp)$ [59], donde n es el conjunto de datos de entrada, c es el número de grupos y p es la dimensión de los datos de entrada. La complejidad aproximada al entrenar dos veces SVM en el peor caso es $O\left(l^3\right) + O\left[\left(\sum_{i=1}^{l} n_i + m\right)^3\right]$. La complejidad total del algoritmo SVM-FCM está dado por

$$O(ncp) + O\left(l^3\right) + O\left[\left(\sum_{i=1}^{l^*} n_i + m\right)^3\right] \tag{5.2}$$

donde l es el número total de grupos, n_i es el número de elementos en el ith grupo, l^* es el conjunto de grupos cuyos centros son vectores soporte y m es el número de elementos en grupos con etiquetas mixtas. Resulta claro que la complejidad algorítmica en la ecuación (5.2) es mucho menor que la complejidad de las SVM clásicas $O(n^3)$.

En el algoritmo propuesto, la complejidad crece con respecto al tamaño de los datos de entrenamiento n y al número de grupos seleccionado. Sin embargo, es claro que el número de vectores soporte obtenido en la primera etapa de clasificación afecta directamente la complejidad del algoritmo. La elección de l es muy importante con el objetivo de obtener un tiempo de entrenamiento rápido. Cuando n es muy grande, el costo de cada iteración será alto y una l muy grande necesita más iteraciones, ya que está convergerá lentamente.

La Figura 5.3 muestra el tiempo de entrenamiento para el conjunto de datos Checkerboard con 40,000 datos de entrenamiento. La Figura 5.3 muestra los diferentes tiempos de entrenamiento al variar el número de grupos. En los experimentos realizados se utilizó

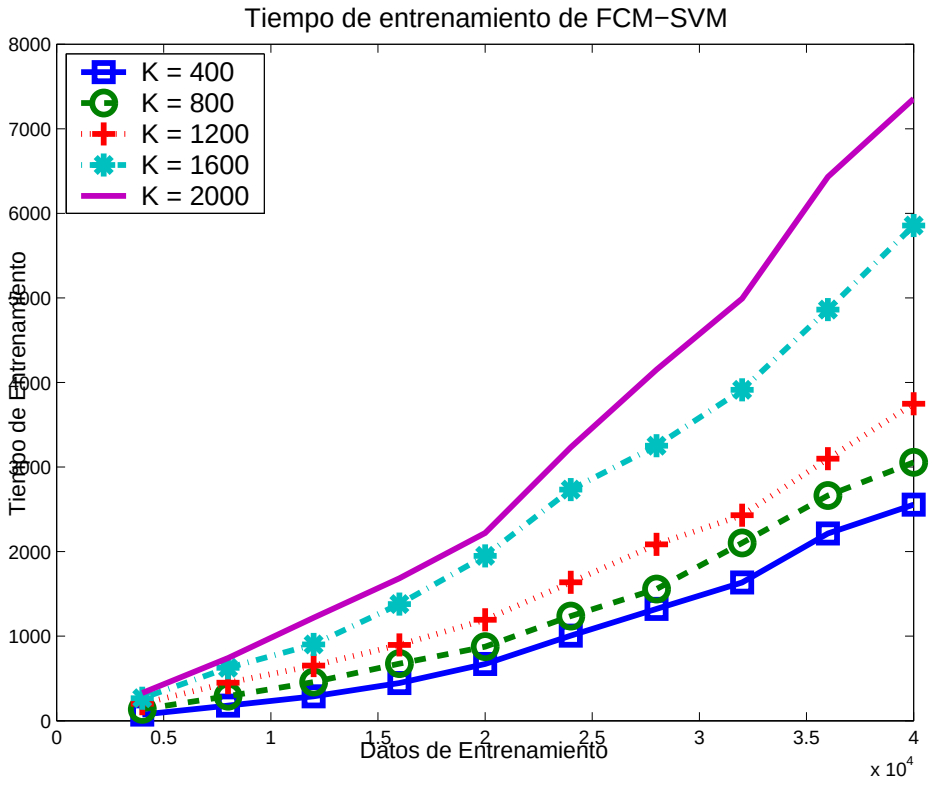

Figura 5.3: Tiempos de entrenamientos con diferente número de grupos (K) con el algoritmo FCM-SVM^2.

un número de grupos entre el 0.5 % y el 1 % del conjunto total de datos de entrada, las precisiones de clasificación obtenidas con el 1 % del total de datos son muy buenas en todos los experimentos realizados. La Figura 5.4 muestra las precisiones de clasificación obtenidas con diferente número de grupos. Sin embargo, para este caso, elegir un número de grupos inicial K menor a 400 afecta considerablemente la precisión del clasificador, debido a que el conjunto de datos tiene demasiados puntos de intersección o fronteras entre clases.

Tiempo de entrenamiento

Debido a que el número de vectores soporte influye directamente en la complejidad

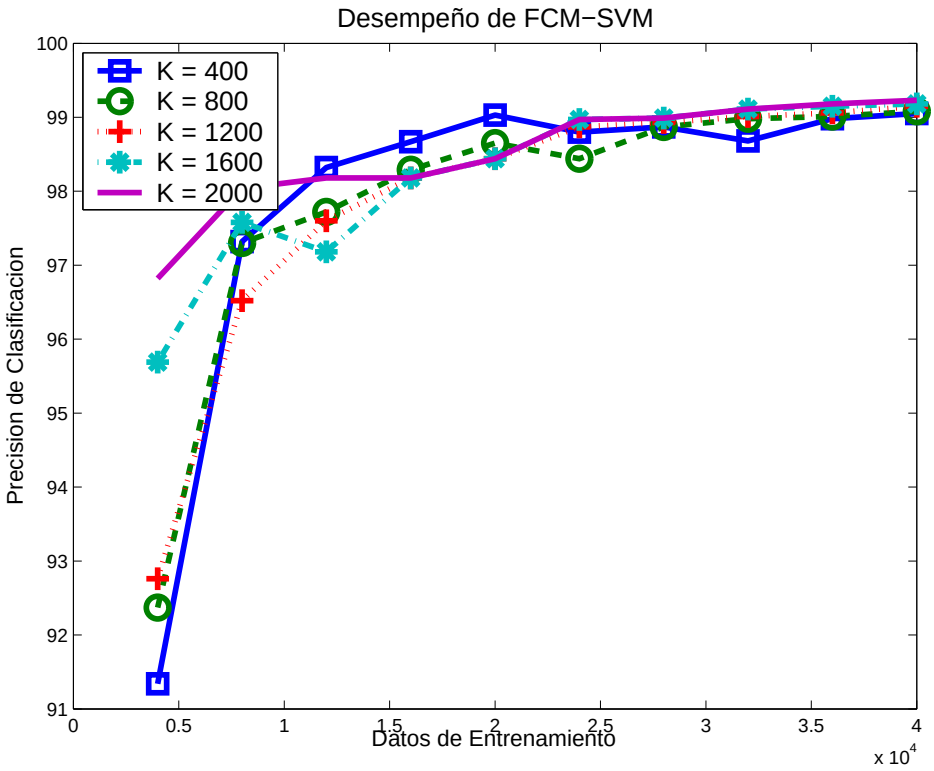

Figura 5.4: Precisión de clasificación de FCM-SVM^2 con diferente número de grupos.

del algoritmo propuesto, en esta subsección se muestra la complejidad del algoritmo con respecto al número de vectores soporte en el conjunto de datos a entrenar. El tiempo de entrenamiento del enfoque propuesto, incluye dos partes en esta Sección: algoritmo de agrupamiento y empleo de SVM. El tiempo de entrenamiento de FCM está dado por

$$T_f = C_n^2 \times c_f = \frac{n \times (n-1)}{2} c_f$$

donde c_f es el costo de evaluar la distancia difusa y n es el número de datos de entrenamiento.

El tiempo de entrenamiento de SVM puede ser calculado de la manera siguiente. Asumimos que el mayor costo computacional está dado por la multiplicación de los op-

eradores (SMO) sin considerar otros costos, tales como acceso a memoria. El parámetro de crecimiento de la probabilidad de que sean vectores soporte se asume que es constante.

Sea $n_m(t)$ el número de puntos que no son vectores soporte en tiempo t. La probabilidad del número de vectores soporte en tiempo t es $F(t)$, la cual está dada por

$$F(t) = \frac{l + m - n_m(t)}{l + m}, \quad n_m(t) = (l + m)\,[1 - F(t)]$$

donde l es el número de centros de grupos y m es el número de elementos en los grupos con etiquetas mixtas. El parámetro de crecimiento del número de vectores soporte (o parámetro de decrecimiento del número de puntos que no son vectores soporte) es

$$h(t) = -\frac{d\,[n_m(t)]}{dt}\frac{1}{n_m(t)} = \frac{\dot{F}(t)}{(l + m)\,[1 - F(t)]}$$

Ya que el parámetro de crecimiento es constante $h(t) = \lambda$, la solución de $\dot{F}$ es

$$\dot{F}(t) = -\lambda\,(l + m)\,F(t) + \lambda\,(l + m)$$

con $F(0) = 0$ es

$$F(t) = 1 - e^{-\lambda(l+m)t}$$

El número de vectores soporte en la primera etapa de clasificación con SVM en tiempo t es $n_{sv1}(t)$, la cual satisface

$$n_{sv1}(t) = (l + m)\,F(t) = (l + m)\,(1 - e^{-\lambda(l+m)t}), \quad \lambda > 0 \tag{5.3}$$

y ésta es monotónicamente creciente.

El número de vectores soporte de la segunda etapa de SVM con tiempo t es $n_{sv2}(t)$, y satisface

$$n_{sv2}(t) = (l_1 + m)\,(1 - e^{-\lambda(l_1+m)t}), \quad \lambda > 0$$

donde

$$l_1 = \sum_{i=1}^{l} n_i + m$$

Se define el número final de vectores soporte en cada grupo de la primera etapa como h_i, $i = 1 \cdots l$. De la ecuación (5.3) tenemos que $h_i = (l+m)\left(1 - e^{-\lambda(l+m)t}\right)$, por lo tanto

$$t_i = \frac{1}{\lambda\,(l+m)} \ln\left(\frac{l+m}{l+m-h_i}\right) \quad i = 1 \cdots l$$

Se define a c_1 como el costo de cada operación de multiplicación para SMO. Para cada paso iterativo, el principal costo es $4\,(l+m)\,c_1$. El costo de la optimización en la primera etapa es

$$\begin{aligned} T_{op}^{(1)} &= 4\,(l+m)\,c_1 \tfrac{1}{\lambda(l+m)} \ln\left(\tfrac{l+m}{l+m-h_i}\right) \\ &= \tfrac{4}{\lambda} c_1 \ln\left(1 + \tfrac{h_i}{l+m-h_i}\right) \\ &\leq \tfrac{4c_1}{\lambda} \tfrac{h_i}{l+m-h_i} \end{aligned}$$

En la segunda etapa,

$$\begin{aligned} T_{op}^{(2)} &= 4\,(l_1+m)\,c_1 \tfrac{1}{\lambda(l_1+m)} \ln\left(\tfrac{l_1+m}{l_1+m-h_i}\right) \\ &\leq \tfrac{4c_1}{\lambda} \tfrac{h_i}{l_1+m-h_i} \end{aligned}$$

Otro costo de cómputo es el cálculo del kernel. Se define c_2 como el costo de evaluar cada elemento del kernel K. En la primera etapa es

$$T_{\text{ker}}^{(1)} = (l+m)\,c_2, \quad T_{\text{ker}}^{(2)} = (l_1+m)\,c_2$$

El tiempo total para el algoritmo SVM-FCM está dado por

$$\text{FCM: } T_1 \leq \tfrac{n\times(n-1)}{2} c_f + \tfrac{4}{\lambda} c_1 \left[\tfrac{h_i}{l+m-h_i} + \tfrac{h_i}{l_1+m-h_i}\right]$$

5.2. Algoritmo MEB-SVM^2

Espacio de memoria

En la primera etapa de agrupamiento, el conjunto entero de datos de entrada

$$X = \{x_1, \ldots, x_n\}, \quad Y = \{y_1, \ldots, y_n\}, \quad y_i = \pm 1, \quad x_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p$$

es cargado en la memoria. Los datos son de tipo flotante, por lo tanto el tamaño de cada datos es 4 bytes. Si se emplea clasificación con SVM clásicas el tamaño de la memoria para los datos de entrada deberá ser $4\left(n \times p\right)^2$. Sin embargo, empleando SMO, el tamaño de la memoria para los datos de entrada deberá ser $4\left(2 \times p\right)^2$, ya que no se carga la matriz en la memoria sino únicamente dos datos en cada iteración, mientras que el tamaño de los datos de agrupamiento es $4\left(n \times p\right)$. El espacio total de agrupamiento con MEB está dado por

$$4\left(2 \times p\right)^2 + 4(n \times p). \tag{5.4}$$

Cuando n es muy grande (grandes conjuntos de datos), n_i, m y $l \ll n$, el espacio de memoria dado por las ecuaciones (5.4) y (5.6) del enfoque propuesto es mucho más pequeño que el espacio de memoria empleado por SVM clásicas $4\left(n \times p\right)^2$.

Complejidad algorítmica

Es claro que sin algoritmo de descomposición, resulta casi imposible para SVM clásicas obtener un hiperplano óptimo cuando el tamaño de los datos de entrenamiento n es demasiado grande. Es difícil analizar de forma precisa la complejidad del algoritmo de SVM. De acuerdo a Steinwart [84], la complejidad del algoritmo SMO es $O(n \cdot n_{sv} + n_{sv}^3)$. Sin embargo, Keerti et al. [41] sostienen que la complejidad esta en el orden de $O(L \cdot n)$.

La complejidad del algoritmo propuesto puede ser calculada de la manera siguiente. La complejidad del agrupamiento usando MEB es $O(\frac{nd}{\in} + \frac{1}{\in^{4,5}} \log \frac{1}{\in})$. La complejidad aproximada al entrenar dos veces SVM es $O\left[(L_1 \cdot l)\right] + O\left[L_2 \cdot \left(\sum_{i=1}^{l} n_i + m\right)\right]$. La complejidad total MEB está dada por

$$O\left(\left(\frac{nd}{\in} + \frac{1}{\in^{4,5}} \log \frac{1}{\in}\right) \times c\right) + O\left[(L_1 \cdot l)\right] + O\left[L_2 \cdot \left(\sum_{i=1}^{l} n_i + m\right)\right] \tag{5.5}$$

donde l es el número total de grupos, n_i es el número de elementos en el ith grupo cuyos centros son vectores soporte, c es el número de grupos y m es el número de elementos en grupos con etiquetas mixtas. Es claro que en la ecuación (5.5), la complejidad algorítmica es mucho menor que la complejidad de las SVM clásicas $O(n^3)$.

En el algoritmo propuesto, la elección de l es muy importante con el objetivo de

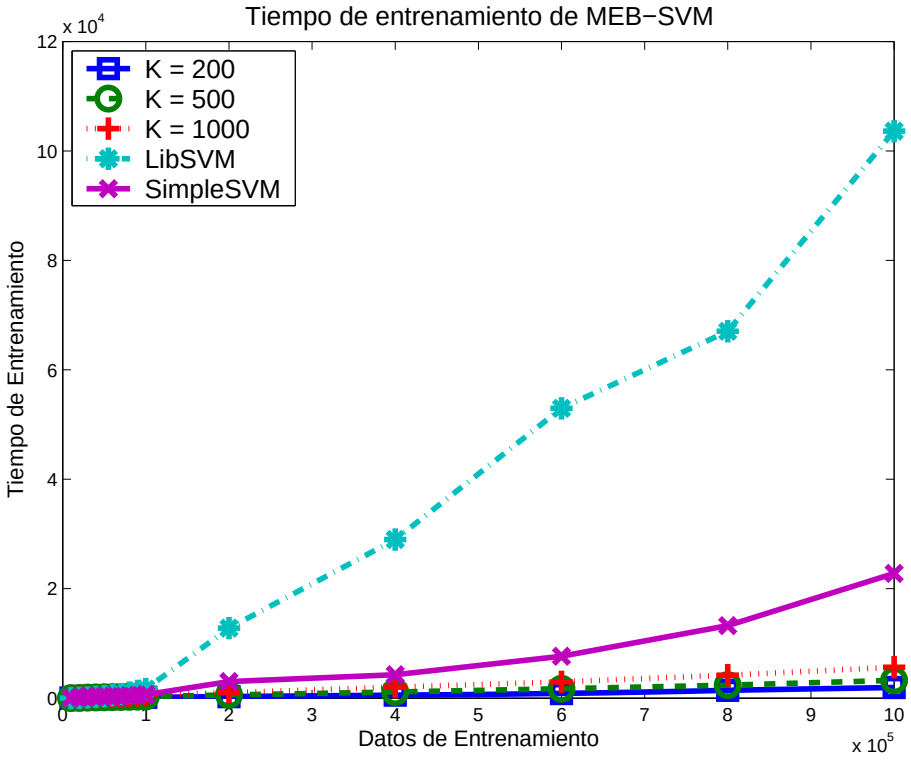

Figura 5.5: Precisión de clasificación con el algoritmo MEB-SVM^2 y otras implementaciones de SVM.

obtener una rápida convergencia. Cuando n es muy grande, el costo computacional será muy alto, sin embargo, si empleamos una l pequeña el algoritmo convergerá rápidamente.

La Figura 5.5 muestra el tiempo de entrenamiento para el conjunto de datos Checkerboard con 1,000,000 datos de entrenamiento con el algoritmo propuesto y en comparación con SimpleSVM y LibSVM. La Figura 5.5 muestra los diferentes tiempos de entrenamiento al variar el número de grupos. En los experimentos realizados se utilizó un número de grupos entre el 0.02 % y el 0.1 % del conjunto total de datos de entrada, las precisiones de clasificación obtenidas con el 0.1 % del total de datos son muy buenas en todos los experimentos realizados. La Figura 5.7 muestra las precisiones de clasificación obtenidas con diferente número de grupos.

En la Figura 5.5, es posible ver las curvas de crecimiento de los tiempos de entrenamiento de los algoritmos SimpleSVM y LibSVM, también se puede apreciar que el tiempo de entrenamiento con el enfoque propuesto es significativamente menor que las implementaciones SimpleSVM y LibSVM, incluso el tiempo de entrenamiento del algoritmo propuesto es tan pequeño en comparación con el de las otras implementaciones que parece una línea recta, pero realmente no lo es. En la Figura 5.6 mostramos únicamente los tiempos de entrenamiento correspondientes al algoritmo propuesto con diferentes grupos de seccionamiento en la primera fase, para ver si su tendencia de crecimiento es similar a la tendencia de crecimiento del algoritmo LibSVM. La Figura 5.6 muestra que las tendencias de crecimiento son similares, la diferencia radica en que el algoritmo LibSVM crece con respecto al número de datos de entrada, mientras que el algoritmo propuesto crece con respecto al conjunto de grupos inicial elegido y es afectado directamente por el número de vectores soporte en el conjunto de datos.

Tiempo de entrenamiento

Debido a que el número de vectores soporte influye directamente en la complejidad del algoritmo propuesto, en esta subsección se muestra la complejidad del algoritmo con respecto al número de vectores soporte en el conjunto de datos a entrenar. El tiempo de entrenamiento del enfoque propuesto incluye dos partes en esta Sección: algoritmo de

agrupamiento y empleo de SVM. El tiempo de entrenamiento de MEB está dado por

$$T_f = \pi \times l \times n \times c_f$$

donde π son las veces de calcular la approximation-$(1+\epsilon)$, l es el número de grupos y c_f es el costo de evaluar la distancia Euclidiana.

El tiempo de entrenamiento de SVM puede ser calculado de la manera siguiente. Se asume que, el mayor costo computacional está dado por la multiplicación de los operadores (SMO) sin considerar el costo de otros operadores, tales como acceso a memoria. El parámetro de crecimiento de la probabilidad de que sean vectores soporte se asume que es constante.

Se define a c_1 como el costo de cada operación de multiplicación para SMO. Para cada iteración, el principal costo es $4\,(l+m)\,c_1$. El costo de optimización en la primera etapa es

$$\begin{aligned} & T_{op}^{(1)} = 4\,(l+m)\,c_1 \tfrac{1}{\lambda(l+m)} \ln\left(\tfrac{l+m}{l+m-h_i}\right) \\ & = \tfrac{4}{\lambda} c_1 \ln\left(1 + \tfrac{h_i}{l+m-h_i}\right) \\ & \leq \tfrac{4c_1}{\lambda} \tfrac{h_i}{l+m-h_i} \end{aligned}$$

En la segunda etapa, es:

$$\begin{aligned} & T_{op}^{(2)} = 4\,(l_1+m)\,c_1 \tfrac{1}{\lambda(l_1+m)} \ln\left(\tfrac{l_1+m}{l_1+m-h_i}\right) \\ & \leq \tfrac{4c_1}{\lambda} \tfrac{h_i}{l_1+m-h_i} \end{aligned}$$

Otro costo de cómputo es el cálculo del kernel. Se define c_2 como el costo de evaluar cada elemento de K. En la primera etapa es

$$T_{\text{ker}}^{(1)} = (l+m)\,c_2, \quad T_{\text{ker}}^{(2)} = (l_1+m)\,c_2$$

El tiempo total para los dos enfoques está dado por

$$\text{MEB: } T_2 \leq \pi \times l \times n \times c_f + \frac{4}{\lambda} c_1 \left[\frac{h_i}{l+m-h_i} + \frac{h_i}{l_1+m-h_i}\right]$$

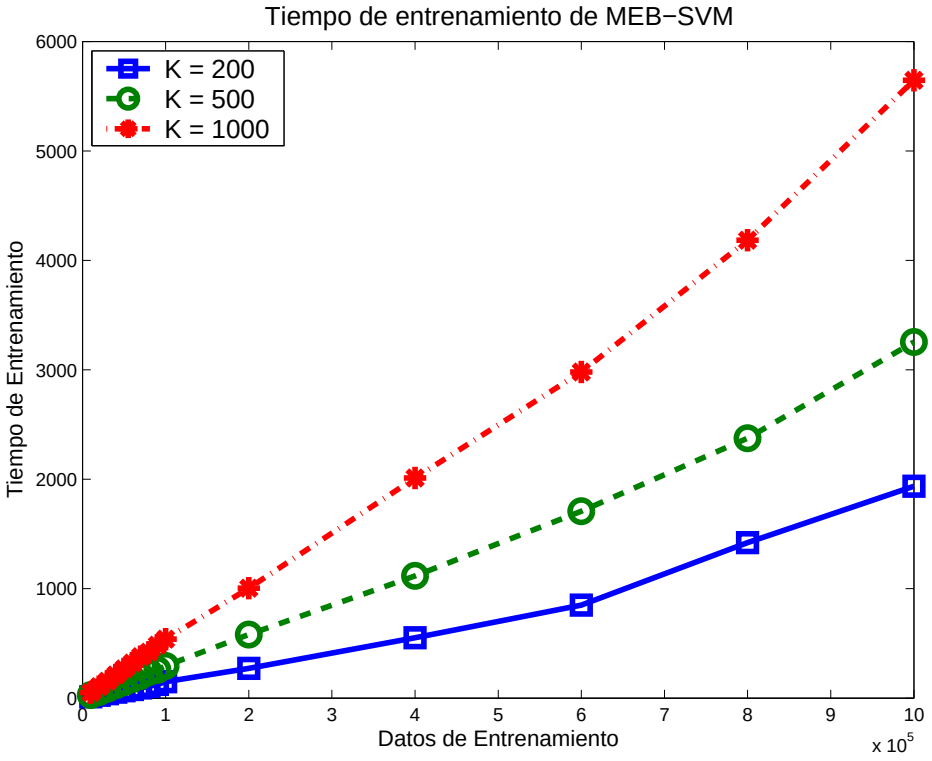

Figura 5.6: Tiempos de entrenamientos con diferente número de grupos (K) con el algoritmo MEB-SVM^2.

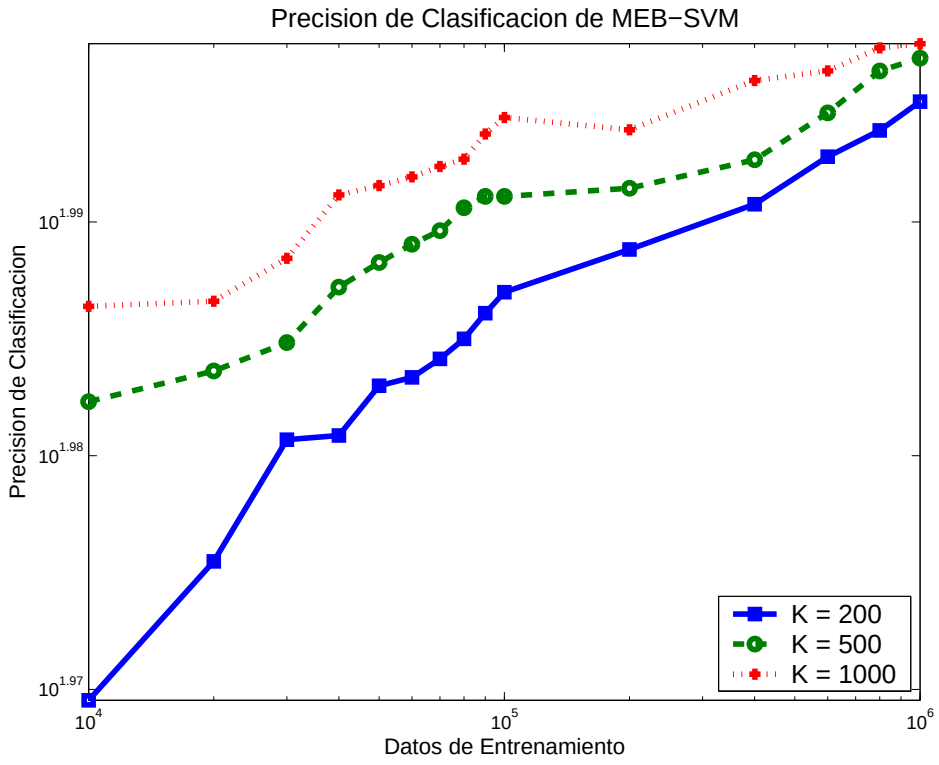

Figura 5.7: Precisiones de clasificación con diferente número de grupos (K) con el algoritmo FCM-SVM^2.

5.3. Algoritmo RS-SVM^2

Espacio de memoria

En la primera etapa de agrupamiento, el conjunto entero de datos de entrada

$$X = \{x_1, \ldots, x_n\}, \quad Y = \{y_1, \ldots, y_n\}, \quad y_i = \pm 1, \quad x_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p$$

es cargado en la memoria. Los datos son de tipo flotante, por lo tanto el tamaño de cada datos es 4 bytes. Si se emplea clasificación con SVM clásicas el tamaño de la memoria para los datos de entrada deberá ser $4\,(n \times p)^2$. Sin embargo, empleando SMO, el tamaño de la memoria para los datos de entrada deberá ser $4\,(2 \times p)^2$, ya que no se carga la matriz en la memoria sino únicamente dos datos en cada iteración, mientras que el tamaño de los datos de agrupamiento es $4\,(n \times p)$. El espacio total de almacenamiento empleando clasificación con SVM basado en selección aleatoria está dado por

$$4\,(2 \times p) + 4(l \times p). \tag{5.6}$$

Cuando n es muy grande (grandes conjuntos de datos), n_i, m y $l \ll n$, el espacio de memoria dado por las ecuaciones (5.4) y (5.6) del enfoque propuesto es mucho más pequeño que el espacio de memoria empleado por SVM clásicas.

Complejidad algorítmica

Es claro que sin algoritmo de descomposición, resulta casi imposible para SVM clásicas obtener un hiperplano óptimo cuando el tamaño de los datos de entrenamiento n es demasiado grande. Es difícil analizar de forma precisa la complejidad del algoritmo de SVM. De acuerdo a Steinwart [84] la complejidad del algoritmo SMO es $O(n \cdot n_{sv} + n_{sv}^3)$. Sin embargo, Keerti et al. [41] sostienen que la complejidad esta en el orden de $O(L \cdot n)$.

La complejidad del algoritmo propuesto puede ser calculada de la manera siguiente. La complejidad del agrupamiento usando MEB es $O(n)$. La complejidad aproximada al entrenar dos veces SVM es $O\left[(L_1 \cdot l)\right] + O\left[L_2 \cdot \left(\sum_{i=1}^{l} n_i + m\right)\right]$. La complejidad total MEB está dada por

$$O\left[(L_1 \cdot l)\right] + O\left(\left(\frac{nd}{\in} + \frac{1}{\in^{4,5}} \log \frac{1}{\in}\right) \times c_{sv}\right) + O\left[L_2 \cdot \left(\sum_{i=1}^{l} n_i + m\right)\right] \tag{5.7}$$

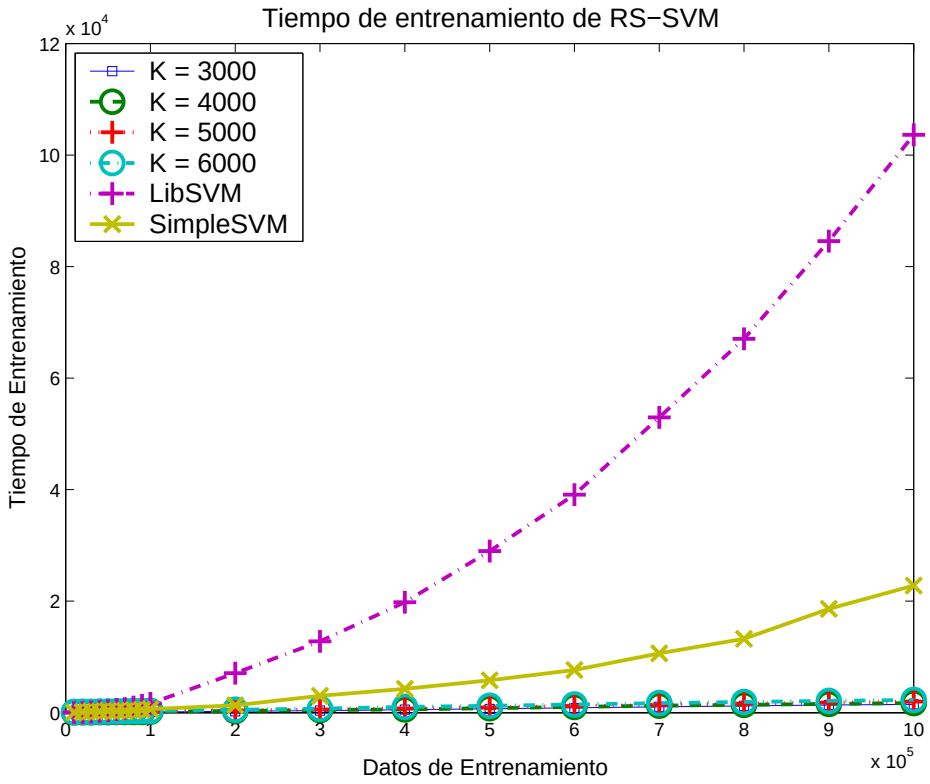

Figura 5.8: Acercamiento a tiempos de entrenamientos con el algoritmo RS-SVM^2 y otras implementaciones de SVM.

donde l es el número total de grupos, n_i es el número de elementos en el ith grupo cuyos centros son vectores soporte, c_{sv} es el número de puntos que son vectores soporte en la primera etapa y m es el número de elementos en grupos con etiquetas mixtas. Es claro que en la ecuación (5.7), la complejidad algorítmica es mucho menor que la complejidad de las SVM clásicas $O(n^3)$.

En el algoritmo propuesto, la elección de l es muy importante con el objetivo de obtener una rápida convergencia. Cuando n es muy grande, el costo computacional será muy alto, sin embargo, si empleamos una l pequeña el algoritmo convergerá rápidamente.

La Figura **??** muestra el tiempo de entrenamiento para el conjunto de datos Checker-

board con 1,000,000 datos de entrenamiento con el algoritmo propuesto y en comparación con SimpleSVM y LibSVM. La Figura 5.9 muestra los diferentes tiempos de entrenamiento al variar el número de grupos. En los experimentos realizados se utilizó un número de grupos entre el 0.3 % y el 0.6 % del conjunto total de datos de entrada, las precisiones de clasificación obtenidas con el 0.6 % del total de datos son muy buenas en todos los experimentos realizados. La Figura 5.9 muestra las precisiones de clasificación obtenidas con diferente número de grupos.

En la Figura **??**, es posible ver las curvas de crecimiento de los tiempos de entrenamiento de los algoritmos SimpleSVM y LibSVM, también se puede apreciar que el tiempo de entrenamiento con el enfoque propuesto es significativamente menor que las implementaciones SimpleSVM y LibSVM, incluso el tiempo de entrenamiento del algoritmo propuesto es tan pequeño en comparación con el de las otras implementaciones que parece una línea recta. En la Figura 5.9 mostramos únicamente los tiempos de entrenamiento correspondientes al algoritmo propuesto para ver si su tendencia de crecimiento es similar a la del algoritmo LibSVM al igual que en el algoritmo anterior. Sin embargo, el tiempo de entrenamiento con RS-SVM^2 es aún menor debido a que el algoritmo no secciona el conjunto de datos de entrada, sólo elije un conjunto de datos inicial de forma aleatoria. La Figura 5.9 muestra que las tendencias de crecimiento son similares, la diferencia radica en que el algoritmo LibSVM crece con respecto al número de datos de entrada, mientras que el algoritmo propuesto crece con respecto al conjunto de grupos inicial elegido y es afectado directamente por el número de vectores soporte en el conjunto de datos.

Tiempo de entrenamiento

Se define el número final de vectores soporte en cada grupo de la primera etapa como h_i, $i = 1 \cdots l$. De la ecuación (5.3) se tiene que $h_i = (l+m)\left(1 - e^{-\lambda(l+m)t}\right)$, por lo tanto

$$t_i = \frac{1}{\lambda\,(l+m)} \ln\left(\frac{l+m}{l+m-h_i}\right) \quad i = 1 \cdots l$$

Se define a c_1 como el costo de cada operación de multiplicación para SMO. Para cada iteración, el principal costo es $4\,(l+m)\,c_1$. El costo de optimización en la primera etapa

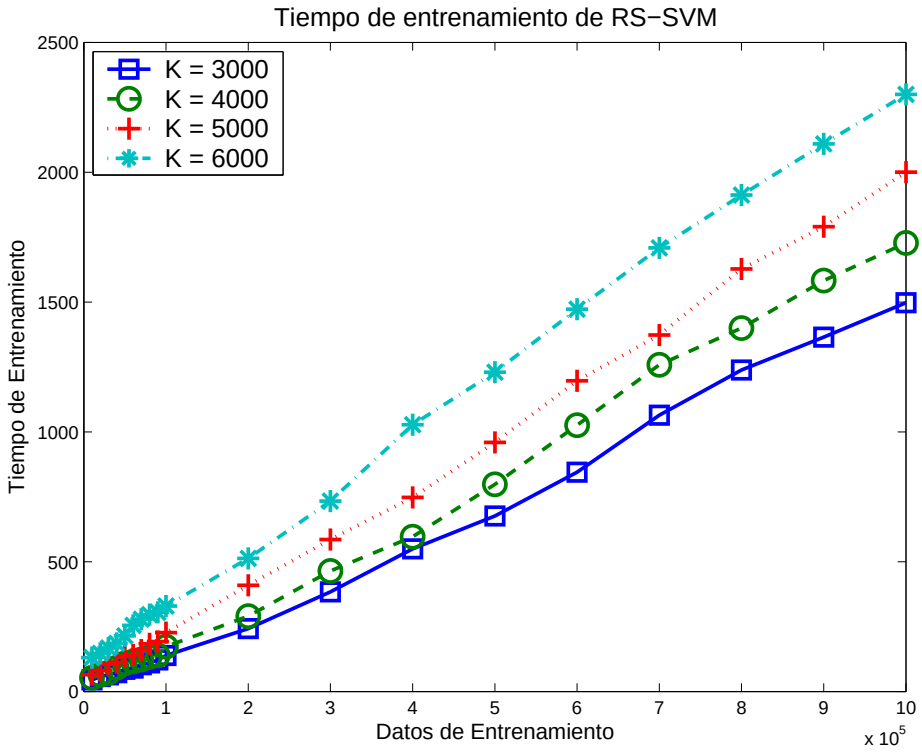

Figura 5.9: Tiempos de entrenamientos con diferente número de puntos inicial con el algoritmo RS-SVM^2.

es

$$\begin{aligned} T_{op}^{(1)} &= 4\,(l+m)\,c_1 \tfrac{1}{\lambda(l+m)} \ln\left(\tfrac{l+m}{l+m-h_i}\right) \\ &= \tfrac{4}{\lambda} c_1 \ln\left(1+\tfrac{h_i}{l+m-h_i}\right) \\ &\leq \tfrac{4c_1}{\lambda} \tfrac{h_i}{l+m-h_i} \end{aligned}$$

En la segunda etapa, es:

$$\begin{aligned} T_{op}^{(2)} &= 4\,(l_1+m)\,c_1 \tfrac{1}{\lambda(l_1+m)} \ln\left(\tfrac{l_1+m}{l_1+m-h_i}\right) \\ &\leq \tfrac{4c_1}{\lambda} \tfrac{h_i}{l_1+m-h_i} \end{aligned}$$

Otro costo de cómputo es el cálculo del kernel. Se define c_2 como el costo de evaluar

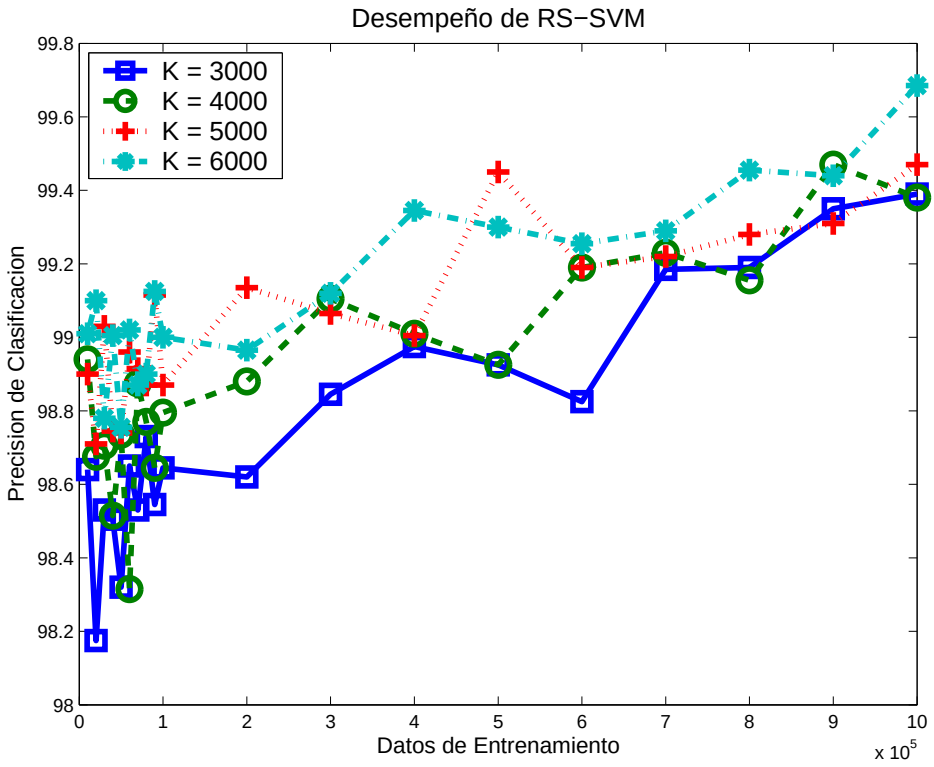

Figura 5.10: Precisiones de clasificación con el algoritmo RS-SVM[2].

cada elemento de K. En la primera etapa es

$$T_{\text{ker}}^{(1)} = (l+m)\, c_2, \quad T_{\text{ker}}^{(2)} = (l_1+m)\, c_2$$

El tiempo total para los dos enfoques está dado por

$$\text{RAN: } T_3 \leq \frac{4}{\lambda} c_1 \left[\frac{h_i}{l+m-h_i} + \frac{h_i}{l_1+m-h_i} \right]$$

5.4. Conclusiones

En este Capítulo, mostramos la complejidad en tiempo y espacio de los tres algoritmos propuestos. La Tabla 5.1 muestra la complejidad algorítmica de los algoritmos

propuestos. Es claro que sin un método de descomposición, resulta casi imposible para las SVM clásicas obtener el hiperplano óptimo cuando el tamaño de los datos de entrenamiento n es muy grande. Entrenar SVM clásicas involucra multiplicación de matrices de tamaño n con complejidad $O(n^{2,3})$ y $O(n^{2,83})$ en el peor caso. Sin embargo, la complejidad del algoritmo SMO depende directamente del número de vectores soporte n_{sv}, un resultado teórico reciente de Steinwart [84] muestra que el número de vectores soporte crece como una función lineal de n. Por lo tanto, para problemas con grandes conjuntos de datos, la complejidad de entrenamiento puede llegar a ser prohibitiva, ya que esta llega a ser $O(n \cdot n_{sv} + n_{sv}^3)$. Sin embargo, Keerti et al. [41] sostienen que la complejidad está en el orden de $O(L \cdot n)$ donde n es el tamaño del conjunto de datos de entrada y L es el número de candidatos a vectores soporte.

En los algoritmo expuestos empleamos SVM clásicas solo en el caso de FCM-SVM^2, la complejidad del algoritmo SVM clásicas es de $O(n^3)$, sin embargo la complejidad del algoritmo está en función del número de grupos y del conjunto de datos reducido, que es conformado por un conjunto de centros de grupos inicial l, un conjunto de elementos que pertenecen a grupos con etiquetas mixtas m y elementos de grupos que son vectores soporte $\sum_{i=1}^{l^*} n_i$. Sin embargo, $\left(l + m + \sum_{i=1}^{l^*} n_i\right) << n$.

En los algoritmos SVM-MEB^2 y SVM-RS^2 empleamos un algoritmo tipo SMO, suponiendo que la complejidad está en el orden de $O(L \cdot n)$ [41]. Para el algoritmo SVM-MEB^2 tenemos que la complejidad está en función del agrupamiento con MEB y las dos fases de SVM, sin embargo la complejidad del agrupamiento con MEB es lineal y la complejidad de las dos fases de SVM está en función de L_1, L_2, l, m y $\sum_{i=1}^{l} n_i$, donde L_1 y L_2 son los candidatos a vectores soporte en la primera y segunda fase respectivamente, resulta claro que $L_1 + L_2 < L$, además el algoritmo SVM-MEB^2 reduce considerablemente el conjunto de datos de entrada a un subconjunto cuyos elementos tienen mayores probabilidades de ser vectores soporte, por lo tanto, $l+m+\sum_{i=1}^{l} n_i << n$ e intuitivamente podemos afirmar que $l + m + \sum_{i=1}^{l} n_i \simeq L$, es claro que la complejidad del algoritmo propuesto está en función de los vectores soporte.

Tabla 5.1: Comparación entre algoritmos propuestos

	Complejidad Algorítmica
FCM-SVM^2	$O(ncp + l^3 + \sum_{i=1}^{l^*} n_i + m^3)$
MEB-SVM^2	$O\left(\left(\frac{nd}{\in} + \frac{1}{\in^{4,5}} \log \frac{1}{\in}\right) \times c + (L_1 \cdot l) + \left(L_2 \cdot \left(\sum_{i=1}^{l} n_i + m\right)\right)\right)$
RS-SVM^2	$O\left(L_1 \cdot l + \left(\left(\frac{nd}{\in} + \frac{1}{\in^{4,5}} \log \frac{1}{\in}\right) \times c_{sv}\right) + \left(L_2 \cdot \left(\sum_{i=1}^{l} n_i + m\right)\right)\right)$

Para el algoritmo SVM-RS^2 tenemos que la complejidad está en función de las dos fases de SVM, ya que el tiempo llevado a cabo en el muestreo aleatorio del conjunto de datos inicial es muy pequeño. Sin embargo, la complejidad de las dos fases de SVM está en función de L_1, L_2, l, m y $\sum_{i=1}^{l} n_i$, donde L_1 y L_2 son los candidatos a vectores soporte en la primera y segunda fase respectivamente. Una vez más, resulta claro que $L_1 + L_2 < L$, además el algoritmo SVM-RS^2 reduce el conjunto de datos de entrada a un subconjunto cuyos elementos tienen mayores probabilidades de ser vectores soporte, por lo tanto, $l + m + \sum_{i=1}^{l} n_i << n$ e intuitivamente podemos afirmar que $l + m + \sum_{i=1}^{l} n_i \simeq L$, es claro que la complejidad del algoritmo propuesto está en función de los vectores soporte i.e. para este caso, la complejidad del algoritmo SVM-RS^2 es reducida al orden $O(L \cdot L)$.

Capítulo 6

Multiclasificación con SVM en dos etapas

Las SVM para clasificación binaria han sido desarrolladas en muchos campos de aplicación. El paradigma de las SVM posee una buena interpretación geométrica discriminando una clase de otra al encontrar un hiperplano, cuyo margen maximiza la separación entre una y otra clase [15][92]. Las SVM han mostrado, en muchos casos, un mejor desempeño sobre otros métodos de clasificación, ya sea redes neuronales, redes Bayesianas o árboles de decisión. Sin embargo, la mayoría de los problemas de clasificación del mundo real poseen conjuntos de datos con pertenencia a más de dos clases. Por ejemplo, para identificar texto puede involucrar varias decenas de clases o para diagnosticar alguna enfermedad asociada a un conjunto de síntomas en una persona puede requerir varias clases, identificar una textura puede requerir varias clases.

Las SVM son métodos de programación cuadrática, cuya fundamentación es la teoría de aprendizaje estadístico de Vapnik [91][92], que permite construir la mejor función $f(x)$ a partir de un conjunto de datos de entrenamiento. La función $f(x)$ está dada por la combinación lineal de algunas funciones base y está depende únicamente de aquellos puntos cercanos a la frontera de clasificación. El nombre de SVM es debido a que la función discriminante lineal obtenida, depende únicamente de este pequeño subconjunto

de datos de entrenamiento conocido como vectores soporte.

Las SVM fueron originalmente diseñadas para problemas de clasificación binaria. Debido a los buenos resultados que han presentado, las SVM han sido extendidas para resolver problemas de clasificación con múltiples clases. Sin embargo, a pesar de los buenos resultados obtenidos al usar SVM en problemas de clasificación binaria, no es trivial extender los algoritmos de clasificación usando SVM, para resolver problemas de clasificación en los que existen múltiples clases. Las SVM no son aptas para trabajar con grandes conjuntos de datos, debido a la complejidad asociada a éstas. Estas desventajas en los problemas de clasificación binaria resultan más acentuadas cuando se trabaja con problemas con múltiples clases, aún en el caso de que ambos tuvieran la misma cantidad de datos de entrenamiento.

Existen dos enfoques principales para problemas de clasificación de múltiples clases. Estos han sido reportados en la literatura como métodos de clasificación para múltiples clases: uno contra todos (OVA -*one against all*-) y uno contra uno (OVO -*one against one*-), que se describen en las siguientes secciones.

6.1. Máquinas de Vectores Soporte Multiclase

La mayor diferencia entre los métodos tradicionales de multi-clasificación con SVM (MSVM) es la forma en como éstas fusionan las salidas de los k clasificadores. Un enfoque para resolver problemas de reconocimiento de patrones de k clasificadores es considerar el problema como una colección de problemas de clasificación binaria. Se construyen k clasificadores, uno para cada clase. El $k - ésimo$ clasificador construye un hiperplano entre la $i - ésima$ clase y las $k - 1$ otras clases. La clase a la que pertenece un punto nuevo es obtenida mediante el argumento máximo obtenido de las k clases. Por otro lado, $\left(\frac{k(k-1)}{2}\right)$ hiperplanos de clasificación son construidos al desarrollar el método uno contra uno. Para determinar la clase a la que pertenece un nuevo punto se cuenta el número de votos que posicionan al nuevo dato en la i-ésima clase. La clase con mayor número de votos será la que se asignará al nuevo dato.

Uno contra todos

Fue la primera implementación para clasificación multiclase con SVM. Este construye k modelos de SVM donde k es el número de clases. La i-ésima SVM es entrenada con todos los ejemplos en la i-ésima clase con etiquetas positivas y los demás con etiquetas negativas, i.e., dados l datos de entrenamiento $(x_1, y_1), \ldots, (x_l, y_l)$, donde $x_i \in R^n, i = 1, \ldots, l$ y $y_i \in \{1, \ldots, k\}$ es la clase de x_i, la i-ésima SVM resuelve el siguiente problema:

$$\min_{w^i, b^i, \xi^i} \frac{1}{2}(w^i)^T w^i + C \sum_{j=1}^{l} \xi_j^i (w^i)^T$$

$$\begin{aligned} (w^i)^T \phi(x_j) + b^i &\geq 1 - \xi_j^i, \quad \text{si } y_j = i \\ (w^i)^T \phi(x_j) + b^i &\leq -1 + \xi_j^i, \quad \text{si } y_j \neq i \\ \xi_j^i &\geq 0, \quad j = 1, \ldots, l \end{aligned} \tag{6.1}$$

donde los datos de entrenamiento x_i son mapeados a un espacio altamente dimensional mediante ϕ, mientras que C es el parámetro de penalización.

Después de resolver la ecuación (6.1), existen k funciones de decisión

$$\begin{gathered} (w^1)^T \phi(x) + b^1 \\ (w^2)^T \phi(x) + b^2 \\ \vdots \\ (w^k)^T \phi(x) + b^k \end{gathered}$$

Se dice que x está en la clase que posee el valor más grande de la función de decisión

$$\text{clase de } x \equiv \arg \max_{i=1,\ldots,k} ((w^i)^T \phi(x_j) + b^i) \tag{6.2}$$

Uno contra uno

Este método construye $k(k-1)/2$ clasificadores, donde cada uno es entrenado sobre datos de dos clases. Para datos de entrenamiento, a partir de la $i - ésima$ y $j - ésima$

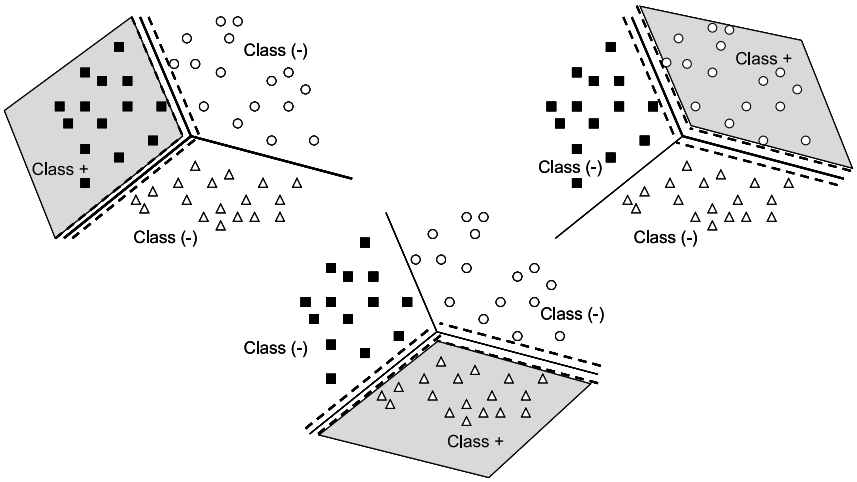

Figura 6.1: Multiclasificación con SVM

clases, el problema de clasificación binaria es dado como:

$$\min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_{j=1}^{l} \xi_t^{ij}(w^{ij})^T$$

$$\begin{aligned} (w^{ij})^T\phi(x_t) + b^{ij} &\geq 1 - \xi_t^{ij}, \quad \text{si } y_t = i \\ (w^{ij})^T\phi(x_t) + b^{ij} &\leq -1 + \xi_t^{ij}, \quad \text{si } y_t \neq j \\ \xi_t^{ij} &\geq 0 \end{aligned} \tag{6.3}$$

Existen diferentes métodos para realizar pruebas futuras después de que los $k(k-1)/2$ clasificadores son construidos. A continuación, se muestran algunas:

1. Si $sign((w^{ij})^T\phi(x) + b^{ij})$ posiciona a x en la i-ésima clase, entonces un voto es adicionado a la i-ésima clase. De otra manera, la j-ésima clase es incrementada en uno, este procedimiento se itera $k(k-1)/2$ veces, al final se posiciona a x en la clase con mayor número de votos. Este enfoque es llamado estrategia de "Máximo Ganador". Cuando se da el caso que dos clases posean idénticos votos, se selecciona aquella con el menor índice.

Al resolver el dual, el número de variables es el mismo al número de datos en dos clases. Aquí, si el promedio de cada clase tiene l/k puntos, se tiene que resolver $k(k-1)/2$ problemas de programación cuadrática, donde cada uno de éstos tiene $2l/k$ variables.

2. Estrategia "Máximo Ganador Modificado". Se aplica una prueba a SVM uno contra uno. Si más de una clase posee idénticos votos, la siguiente estrategia es usada:

   Supóngase que existen p clases de las cuales los máximos votos son igual a m. Cada una de las p clases tiene m funciones $f_y(x), i = 1, \ldots, p, j = 1, \ldots, m$. En este caso, el atributo de clase de x es determinado por la suma de las máximas distancias al hiperplano de clasificación óptima, i.e., la función de decisión auxiliar es

   $$d(x) = \arg \max_{\substack{i \in \{h_1, \ldots, h_p\} \\ \{h_1, \ldots, h_p\} \subseteq \{1, \ldots, k\}}} \left\{ \sum_{j=1}^{m} |f_y(x)| \right\}$$

   La función es definida como

   $$f_i = \max \{ f_{i1}, \ldots, f_{iN} \}, i = 1, \ldots, k$$

   donde $f_y(x) = sign((w^{ij})^T \phi(x) + b^{ij})$

3. Estrategia de "Probabilidades". El autor [52] sugiere utilizar una función sigmoide adicional para estimar probabilidades

   $$P(w_i = 1 \mid x) = \frac{1}{1 + \exp(Af(x) + B)}$$

   donde los parámetros A y B son derivados minimizando la probabilidad negativa de los datos de entrenamiento. Sin embargo, nada garantiza que la suma de todas las probabilidades $\sum_{j=1}^{c} P(w_i \mid x)$ sea igual a 1.

Grafo Acíclico Dirigido (DAGSVM)

Otro método de entrenamiento multiclase con SVM es el Grafo Acíclico Dirigido (*Directed Acyclic Graph Support Vector Machines -DAGSVM-*). Éstas fueron propuestas

por Platt, Christianini y Shawe-Taylor [68], el algoritmo fue propuesto con el objetivo de resolver el problema de regiones no clasificables durante la fase de entrenamiento de SVM con multiples clases. El método de entrenamiento es similar al método uno contra uno, resolviendo $k(k-1)/2$ SVM binarias. Sin embargo, en la fase de prueba, DAGSVM emplea un grafo acíclico dirigido binario con $k(k-1)/2$ nodos internos y k ramificaciones. Cada nodo es una SVM binaria de la $i-ésima$ y la $j-ésima$ clases.

Dado un conjunto de prueba x, iniciando en la raíz del nodo, la función de decisión binaria es evaluada, entonces éste se mueve a la izquierda o a la derecha dependiendo del valor de salida. De esta manera, se recorre un camino antes de alcanzar la ramificación que indica la clase a la que pertenece. La gran ventaja de *DAGSVM* radica en que el proceso de clasificación es más rapido que los métodos convencionales descritos anteriormente.

6.2. Algoritmo de SVM multiclase en dos etapas (mMEB-SVM^2)

El problema de clasificación de k clases radica en construir una función de decisión, a partir de l ejemplos independientes e idénticamente distribuidos $(x_1, y_1), \ldots, (x_l, y_l)$ de una función desconocida, donde $x_i, i = 1, \ldots, l$ es un vector con longitud d y $y_i \in \{1, \ldots, k\}$ representa la clase del ejemplo.

La función de decisión $f(x, \alpha)$ que satisface un punto x, es elegida a partir de un conjunto de funciones definidas por el parámetro α. La tarea consiste en elegir el parámetro α de tal forma que, para cualquier ejemplo que no haya sido empleado durante la fase de entrenamiento, la función obtenida proporcione una etiqueta de clasificación y tan cercana a la función como sea posible.

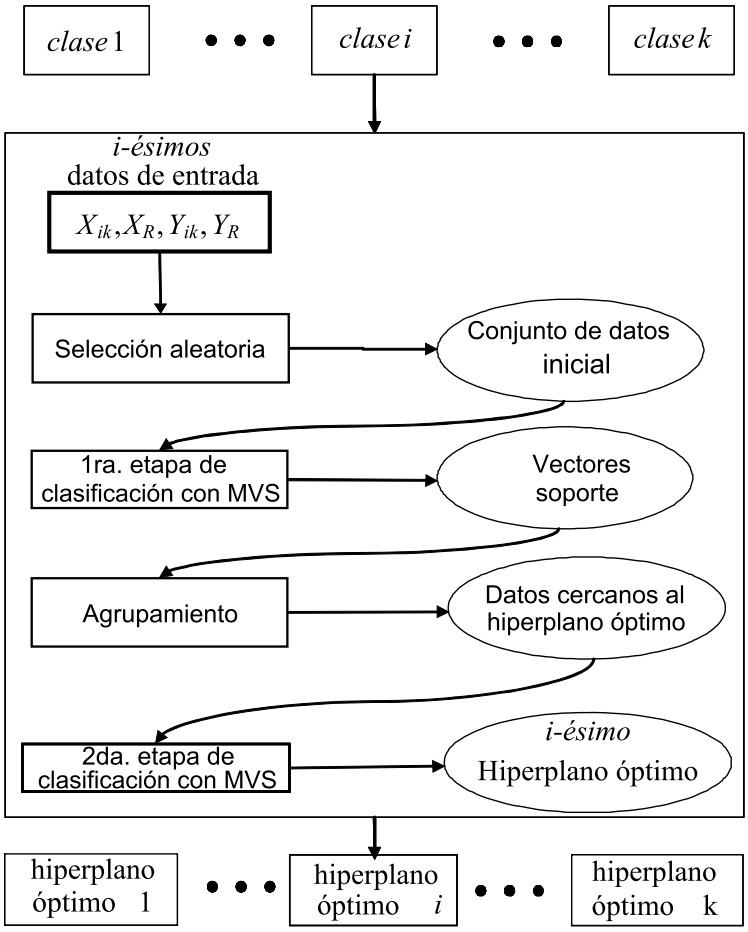

Figura 6.2: Clasificacion con SVM en la $i-esima$ clase

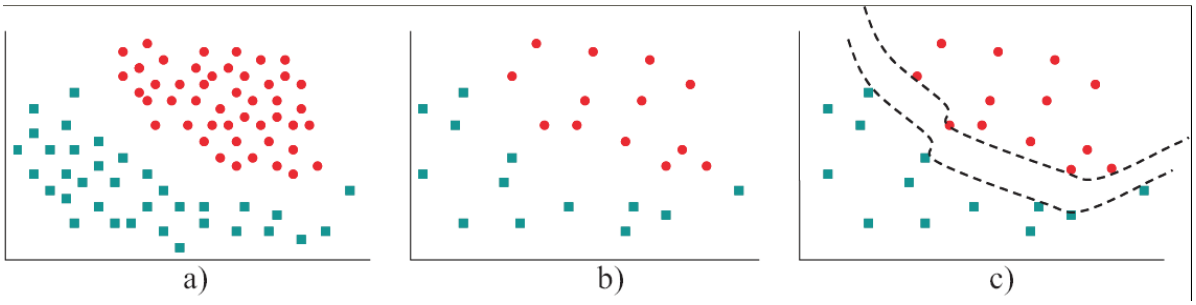

Figura 6.3: Tres etapas del algoritmo. a) Conjunto de datos inicial, b) Conjunto de datos elegido aleatoriamente y c) Clasificacion con SVM sobre el conjunto de datos seleccionado aleatoriamente

6.2.1. Algoritmo de Clasificación Binaria

La Fig. 6.2 muestra el algoritmo de clasificación binaria con múltiples iteraciones. En primera instancia, se selecciona un número de datos a partir del conjunto de datos original empleando un método de selección aleatoria. Estos datos seleccionados son usados en una primera etapa de SVM para obtener un conjunto de vectores soporte iniciales. Una vez concluido este paso, el algoritmo localiza los vectores soporte más cercanos con diferente etiqueta de clase y construye una esfera a partir de cada par de vectores soporte con diferente etiqueta empleando el método de Esferas de Cerradura Mínima. Una vez empleado MEB, todos los puntos encerrados en las esferas son obtenidos. A este proceso se le llama *recuperación de datos*; estos datos recuperados constituyen la distribución de clases, Finalmente, la segunda etapa de clasificación empleando SVM es utilizada sobre el conjunto de datos recuperado para obtener resultados de clasificación más precisos.

1) Selección de datos iniciales

Después de la selección aleatoria, se denota al conjunto de datos muestreado como C. Es posible dividir C en dos subconjuntos, uno conteniendo etiquetas positivas y el

otro conteniendo etiquetas negativas, i.e.,

$$\begin{aligned} C^+ &= \{\cup C_i \mid y = +1\} \\ C^- &= \{\cup C_i \mid y = -1\} \end{aligned} \tag{6.4}$$

La Fig. 6.3-a) muestra el conjunto de datos original, donde los círculos representan los datos con etiquetas positivas, mientras que los cuadros representan los datos con etiquetas negativas. Después de la selección aleatoria, se obtiene un conjunto de datos muestreado a partir del conjunto de datos original. Como se puede apreciar en la Fig. 6.3-b). el conjunto C^+consiste de aquellos datos representados por círculos Fig. 6.3-b), mientras que el conjunto C^-consiste de aquellos datos representados por cuadros en la Fig 6.3-b). Únicamente estos datos serán empleados como datos de entrenamiento en la primera etapa de clasificación con SVM.

2) La primera etapa de clasificación con SVM

En la primera etapa de clasificación con SVM se emplea el algoritmo SMO con el objetivo de obtener el hiperplano de clasificación:

$$\sum_{k \in V_1} y_k \alpha^*_{1,k} K\left(x_k, x\right) + b^*_1 = 0 \tag{6.5}$$

donde V_1 es el conjunto de vectores soporte obtenido a partir del conjunto de datos muestreado y de la primera etapa de clasificación con SVM. El conjunto de datos de entrenamiento es $C^+ \cup C^-$, obtenido mediante selección aleatoria. La Fig. 6.3 b) y c) ilustran los datos de entrenamiento y los vectores soporte obtenidos en la primera etapa de clasificación, respectivamente.

3) Recuperación de datos

Es fácil notar que, el tiempo de entrenamiento inicial es muy pequeño, ya que el conjunto inicial muestreado aleatoriamente es muy pequeño con respecto al conjunto de datos original. En la primera etapa de clasificación, el conjunto de datos de entrenamiento

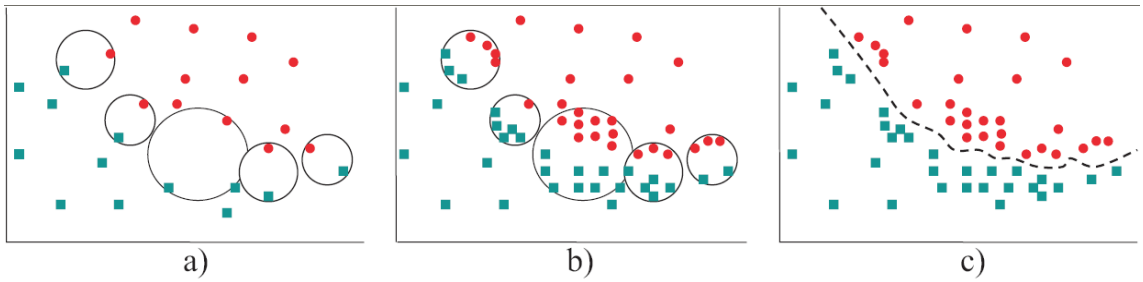

Figura 6.4: Tres etapas del algoritmo. a) Selección de vectores soporte más cercanos con diferente etiqueta de clase, b) obtención de todos los puntos dentro de los vectores cercanos obtenidos y c) Clasificación con SVM sobre el conjunto de datos reducido

es únicamente un pequeño porcentaje del conjunto de datos original. Es claro que esto puede afectar la precisión de clasificación, i.e., el hiperplano de decisión obtenido puede no ser lo suficientemente preciso, siendo necesario un proceso de refinamiento. Aunque el hiperplano obtenido no nos dá una precisión de clasificación muy buena, nos dá por lo menos una referencia sobre qué datos es posible eliminar sin afectar la precisión de clasificación. Es claro que muchos datos útiles no han sido seleccionados durante la etapa de selección aleatoria. Por lo tanto, una idea natural es recuperar aquellos datos que se encuentran localizados entre dos vectores soporte con diferente etiqueta y emplear los datos recuperados para entrenar una SVM una vez más y obtener un mejor hiperplano de clasificación.

Primeramente, con el objetivo de recuperar los datos se propone emplear un conjunto de pares de vectores soporte con diferente etiqueta, donde cada par elegido tiene una *máxima distancia permitida (mdp)* entre cada par. Aquellos pares de vectores soporte que sobrepasan *mdp* no son utilizados para recuperar datos, únicamente aquellos con una distancia entre vectores soporte con diferente clase menor a *mdp*. De esta forma, el algoritmo encuentra la *mínima distancia (mind)* y la *distancia máxima (maxd)* entre dos vectores soporte con diferente etiqueta y finalmente calcula la *mdp*. En este Capítulo, con el objetivo de seleccionar únicamente los pares de vectores más cercanos, se emplea

la siguiente ecuación para calcular *mdp*.

$$x_{ij} = \left\| sv_i^P - sv_j^N \right\| \tag{6.6}$$

$$mdp = \frac{\sum_{i=1}^{n_r} x_{ij}}{n_r} \forall x_{ij} > \bar{x} \tag{6.7}$$

donde n_r es el número de elementos $x_{ij} < \bar{x}$.

$$\bar{x} = \frac{\text{mín}\, d + \text{máx}\, d}{2} \tag{6.8}$$

Una vez obtenida la mínima distancia permitida, se construyen las esferas empleando BCM a partir de cada par de vectores soporte y se emplean estas esferas con el objetivo de obtener todos los datos cercanos al hiperplano contenidos en el conjunto de datos original. Las Figuras. 6.4 a) y 6.4 b) ilustran el proceso de recuperación de datos. Los datos dentro de los círculos mostrados en la Fig. 6.4 b) son los datos recuperados. La cantidad de datos recuperados depende de la selección de la *mdp* y en este caso es controlada por los pares de vectores soporte seleccionados.

Luego, el conjunto de datos recuperados está dado por $\cup_{C_i \in V} \{B_i (c_i, r)\}$.

4) La segunda etapa de clasificación con SVM

Tomando los datos recuperados como un nuevo conjunto de datos de entrenamiento, se usa una vez más clasificación con SVM para obtener el hiperplano de decisión final

$$\sum_{k \in V_2} y_k \alpha_{2,k}^* K(x_k, x) + b_2^* = 0 \tag{6.9}$$

donde V_2 es el conjunto de vectores soporte obtenido en la segunda etapa de clasificación. Generalmente, el hiperplano referido en la ecuación (6.5) es cercano al hiperplano referido en la ecuación (6.9).

Resulta claro que el tiempo de entrenamiento de la primera etapa de clasificación es muy pequeño ya que el conjunto de datos es muy pequeño con respecto al original. Además, los datos recuperados están conformados por los datos más importantes desde

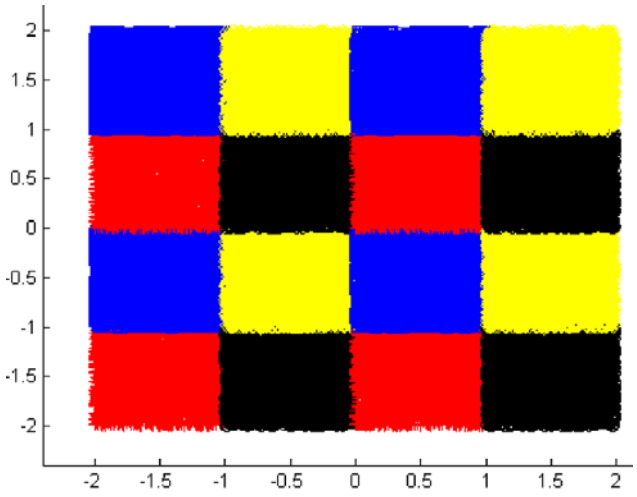

Figura 6.5: Conjunto de datos Checkerboard 4×4

el punto de vista de optimización, ya que los datos más alejados del hiperplano son eliminados. Por otro lado, la mayoría de los datos cercanos al hiperplano referido en la ecuación (6.5) son incluidos mediante la técnica de recuperación de datos. El hiperplano referido en la ecuación (6.9) es una aproximación de un hiperplano obtenido mediante el algoritmo SMO con todos los datos de entrenamiento. La Figura 6.4 (c) muestra los datos de entrenamiento y el hiperplano en la segunda etapa de clasificación con SVM.

6.2.2. Multiclasificación con SVM

Dado un conjunto de datos de entrenamiento inicial $X = \{x_1, x_2, \ldots, x_n\}$, con sus respectivas etiquetas de clase $Y = \{y_1, y_2, \ldots, y_n\}$, el conjunto de datos de prueba es definido como $X_t = \{x_{k1}, x_{k2}, \ldots, x_{kn}\}$, y las etiquetas de clase del conjunto de prueba como $Y_t = \{y_{k1}, y_{k2}, \ldots, y_{kn}\}$, donde k es el número de clases. Una vez definido el algoritmo de clasificación binaria, el algoritmo de multiclasificación queda de la siguiente forma

Paso 1. Obtener los k hyperplanos óptimos empleando el algoritmo de clasificación

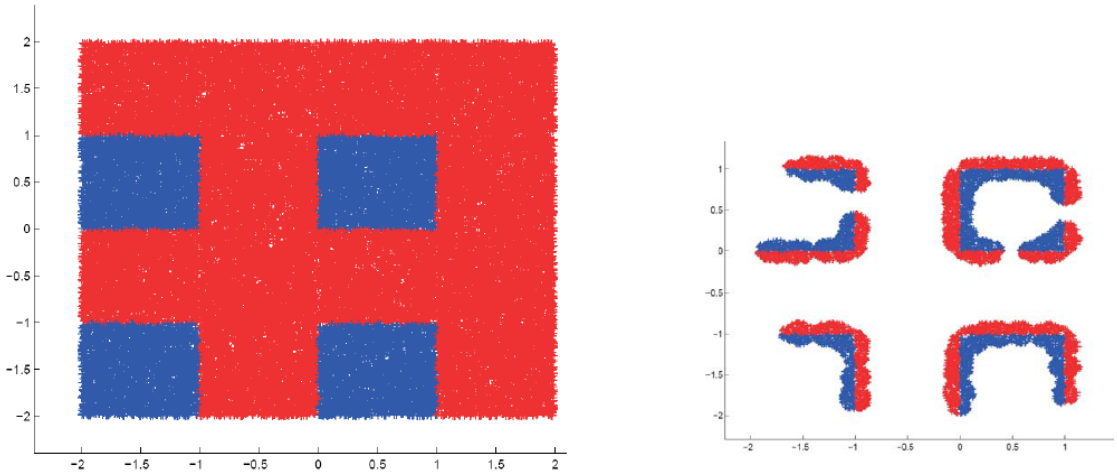

Figura 6.6: Dos etapas del algoritmo en el entrenamiento de la i-ésima clase a) conjunto de datos inicial y b) conjunto de datos reducido

binaria

Paso 2. Dado un ejemplo x para clasificar, su etiqueta de clase es encontrada mediante la siguiente ecuación:

$$clase\ de\ x = \arg \max_{i,2,\ldots,k} (w_i \cdot x_t + b_i)$$

donde w_i y b_i describen el hiperplano de clasificación de la i-ésima clase.

6.3. Resultados experimentales

El algoritmo propuesto fue evaluado sobre 4 diferentes conjuntos de datos. 1) El conjunto de datos 4×4 checkerboard, 2) el conjunto de datos Shuttle, 3) el conjunto de datos Covtype, 4) y el conjunto de datos Sensit Vehicle (combined). Los resultados obtenidos con el algoritmo propuesto fueron comparados con las implementaciones de SVM; Simple SVM [96] y LIBSVM [9] basada en SMO, todos los conjuntos de datos están disponibles en [9].

Tabla 6.1: Resultados de comparación conjunto de datos Checkerboard, t = tiempo en segundos, Acc = precisión

Conjunto de datos **Checkerboard**						
	Enfoque Propuesto		LIBSVM		Simple SVM	
#	**t**	**Acc**	**t**	**Acc**	**t**	**Acc**
50,000	93.6	99.9	2097	99.9	109	99.9
100,000	145	99.9	9441	99.9	257	99.9
500,000	316	99.9	72893	99.9	685	99.9
1,000,000	517	99.9	198,300	99.9	4580	99.9

#= tamaño del conjunto de datos, t=tiempo en segundos, Acc=precisión.

Con el propósito de clarificar y enfatizar la idea básica del enfoque propuesto, consideramos el caso más simple de clasificación y seccionamiento.

Ejemplo 6.1 *El conjunto de datos* 4×4 *checkerboard es comúnmente usado para evaluar implementaciones de SVM con grandes conjuntos de datos. En este caso el conjunto de datos de entrada tiene cuatro clases como se muestra en la Figura 6.5. Para mostrar el desempeño del algoritmo propuesto con respecto al tamaño del conjunto de datos de entrada y tiempo de entrenamiento, se emplean conjuntos de* $1,000,000$, $500,000$, $100,000$ *y* $50,000$ *de forma separada en los experimentos, mientras que un conjunto con* $5,000$ *datos es empleado para prueba.*

Después de aplicar el muestreo aleatorio, la primera etapa de SVM y el algoritmo de MEB sobre los vectores soporte más cercanos con diferente etiqueta de clase. Como se puede apreciar en la Figura 6.6 el conjunto de datos original con $500,000$ datos (Figura 6.6 *a*)) es reducido considerablemente a únicamente 7291 datos como se muestra en la Figura 6.6 *b*)

La Tabla 6.1 muestra los resultados de los experimentos realizados. En los experimentos se utilizó el kernel RBF en todas las implementaciones de SVM. Para el conjunto de datos con $50,000$ datos de entrada, el algoritmo propuesto necesitó 93 segundos, mientras que LIBSVM y Simple SVM necesitaron alrededor de 4600 y 110 segundos respectivamente, mientras que las precisiones son muy similares. Para $1,000,000$ datos de entrenamiento, el enfoque propuesto requirió 517 segundos, mientras que para LIBSVM, y Simple SVM el tiempo de entrenamiento fue muy grande.

Ejemplo 6.2 *El conjunto de datos Shuttle consiste de* 7 *clases. El conjunto de datos contiene* $43,500$ *datos con 9 características cada uno. En los experimentos realizados se usaron conjuntos de datos con* $22,000$ *y* $43,500$ *datos de entrenamiento, mientras que para probar los resultados obtenidos se utilizó un conjunto de datos de prueba con* $14,500$ *datos. En los experimentos realizados se usó el kernel RBF con* $\gamma = 0{,}5$.

La Tabla 6.2 muestra los resultados obtenidos y la comparación de éstos con otras implementaciones de SVM en tiempo de entrenamiento y precisión de clasificación.

En los experimentos realizados con el Ejemplo 6.2, se usó kernel RBF con $C = 1 \times 10^5$. Los experimentos realizados con el algoritmo propuesto sobre el conjunto de datos con $22,000$ registros requiere alrededor de 15 segundos y LIBSVM requiere cerca de 50 segundos, mientras que las precisiones de clasificación son muy similares. Sin embargo, cuando se trabajó con el conjunto de datos de $43,500$ registros, el clasificador propuesto emplea alrededor de 21 segundos en la fase de entrenamiento, mientras que LIBSVM y Simple SVM requieren demasiado tiempo para entrenar el conjunto de datos.

Ejemplo 6.3 *El conjunto de datos Covtype consiste de* $581,012$ *datos, cada dato pertenece a una de 7 clases con 54 características cada uno. En los experimentos realizados se emplearon conjuntos de datos con* $10,000$, $25,000$, $50,000$, $100,000$ *y* $200,000$ *registros para entrenamiento y* $381,012$ *registros para prueba.*

En los experimentos realizados con el conjunto de datos de $10,000$ registros, el clasificador propuesto requiere 13.7 segundos en la fase de entrenamiento mientras que LIBSVM emplea 328 segundos. La Tabla 6.3 muestra los resultados obtenidos. Es claro que

Tabla 6.2: Resultados del conjunto de datos Shuttle

Conjunto de datos *Shutle*

	Enfoque propuesto		Libsvm		Simple SVM	
#	t	Acc	t	Acc	t	Acc
22000	14.9	99.6	50.5	99.6	60.0	99.6
43500	20.7	99.6	158.	99.7	165	99.8

#= tamaño del conjunto de datos, t=tiempo en segundos, Acc=precisión.

la precisión de clasificación se ve afectada al disminuir el tamaño del conjunto de datos de entrenamiento original. Sin embargo, ya que la mayoría de los datos eliminados son los menos representativos del conjunto de datos original, la precisión de clasificación final obtenida mediante el algoritmo propuesto y otras implementaciones de SVM es muy similar. Con el conjunto de datos entero (25000 registros) el clasificador propuesto emplea 18.3 segundos, mientras que LIBSVM tarda demasiado tiempo en entrenar el conjunto de datos. Sin embargo, una vez más la precisión de clasificación es casi la misma. En los experimentos realizados se utilizó $\gamma = 0{,}75$ y $C = 100000$.

Ejemplo 6.4 *El conjunto de datos Sensit Vehicle (combinado) consiste de* $78,823$ *datos de entrenamiento y* $19,705$ *datos de prueba. Cada dato pertenece a una de tres clases. Cada registro del conjunto de datos tiene 100 características. En los experimentos realizados se utilizaron conjuntos de datos de entrenamiento con* 9000*,* $18,500$*,* $39,000$ *y* $78,823$ *registros.*

Para el conjunto de datos entero (78,823 registros) el clasificador propuesto emplea tiempos de entrenamiento menores con respecto a LIBSVM. Por otro lado, no existe una gran diferencia entre las precisiones de clasificación obtenidas con el algoritmo propuesto y otras implementaciones de SVM. Algunos resultados no son incluidos en la Tabla 6.4 debido al enorme tiempo de entrenamiento empleado en entrenar el conjunto de datos.

Tabla 6.3: Resultados de entrenamiento del conjunto de datos CovType

Conjunto de datos *CovType*

	Enfoque propuesto		Libsvm		Simple SVM	
#	t	Acc	t	Acc	t	Acc
10000	13.7	67.8	328.3	68.1	16598	68.7
25000	18.3	67.9	3104	70.2	57,378	68.9
50000	35.7	69.3	9691	70.3	129,371	70.2
100000	93.9	70.1	37388	70.4	-	-
200000	651	70.3	82739	70.7	-	-

= tamaño del conjunto de datos, t=tiempo en segundos, Acc=precisión.

Tabla 6.4: Resultados de entrenamiento del conjunto de datos Vehicle

Conjunto de datos *Vehicle*

	Enfoque propuesto		Libsvm		Simple SVM	
#	t	Acc	t	Acc	t	Acc
9000	23.7	82.7	56.2	83.3	19108	83.2
18500	31.8	83.9	469	84.6	62375	83.7
39000	69.7	84.2	1842	84.6	-	-
78823	258	84.6	4271	84.8	-	-

6.4. Conclusiones

En este Capítulo, se presentó un enfoque para clasificar grandes conjuntos de datos cuando las etiquetas de clase son múltiples mMEB-SVM^2. El tiempo de entrenamiento de los algoritmos de multiclasificación con SVM resulta ser muy grande en comparación con el tiempo de entrenamiento de algoritmos de clasificación binaria con SVM, aún cuando el tamaño del conjunto de los datos sea el mismo. Es claro que son necesarias técnicas que reduzcan el tiempo de entrenamiento de multiclasificación con SVM.

El algoritmo propuesto emplea un algoritmo previo para detectar la distribución de clases. Con el objetivo de reducir el conjunto de datos de entrenamiento y con ello el tiempo de entrenamiento de las SVM, a partir de las distribuciones de clases se obtienen los puntos más representativos del conjunto de datos entero y se eliminan los menos representativos, para ello se emplea un algoritmo basado en MEB. La diferencia del algoritmo propuesto con el algoritmo MEB-SVM^2, radica en que el algoritmo de recuperación en restringido a un espacio muy pequeño que acotado por pares de vectores soporte con diferente etiqueta de clasificación. Además, al buscar el algoritmo datos entre los vectores soporte opuestos de la primera fase, el algoritmo refina el hiperplano de manera más eficiente que los algoritmos propuestos en el Capítulo 3. En este Capítulo se comparan los resultados obtenidos con dos implementaciones de SVM para grandes conjuntos de datos LibSVM y SimpleSVM. Los experimentos realizados con grandes conjuntos de datos artificiales y reales, muestran que la técnica propuesta tiene una gran ventaja cuando los conjuntos de datos de entrada son muy grandes.

Capítulo 7

Aplicación en biología

En todas las areas de investigación médica y biológica, el rol de las computadoras ha crecido dramáticamente en los últimos años. El uso de las computadoras ha permitido capturar, almacenar, analizar e integrar información médica o biológica de una forma más fácil. La Bioinformática representa una nueva y creciente area de investigación que puede ser definida como la aplicación de enfoques y tecnología computacional para resolver problemas y manejar información biológica. La bioinformática ha concentrado en los últimos años una gran atención en el proyecto del Genoma Humano abarcando el estudio de información genética, estructuras moleculares, funciones bioquímicas y síntomas fenotípicos expuestos. Las herramientas computacionales han llegado a ser esenciales en diversos campos de aplicación biológica, que van desde clasificación de secuencias, detección de similaridades, separación de proteínas de regiones que codifican a partir de regiones que no codifican en secuencias de acido desoxirribonucléico -ADN- (*splicing*), predicción de estructuras moleculares, así como el descubrimiento de nuevos medicamentos y terapias. Las enormes cantidades de datos empleados en los experimentos biológicos actuales han provocado la creación de muchas bases de datos que contienen genomas, secuencias de proteínas y otros tipos de datos biológicos. Todo esto ha obligado a que la investigación básica actual en Bioinformática esté orientada al diseño e implementación de sistemas que permitan resolver problemas de almacenaje, manejo, procesamiento y

análisis de enormes cantidades de ADN.

Un importante problema en Bioinformática es la identificación de genes dentro de grandes regiones de secuencias de ADN. El reconocimiento de genes no es un reto fácil debido a las grandes cantidades de secuencia intergénica y al hecho de que regiones que codifican en proteínas (exones) pueden ser interrumpidas por regiones que no codifican o regiones basura (intrones). Las secuencias que interrumpen los genes de acido ribonucleico (RNA) y proteínas están caracterizadas, sin embargo no son claramente definidas mediante características locales en los empalmes de cada unión. Identificar intrones dentro de cadenas de ADN presenta un gran reto computacional. En algunos organismos los intrones son pocos y los sitios de empalmes son bien caracterizados. No obstante, en algunas otras secuencias, incluyendo el genoma humano, es un gran problema localizar la correcta transición entre las regiones que codifican y las que no, y por lo tanto es un reto determinar la secuencia de mRNA a partir de secuencias de ADN. Aunado a esto, los genes en muchos organismos empalman de diferente manera, dependiendo del tipo de tejido o etapa de desarrollo, complicando la tarea considerablemente.

Algunas herramientas como redes neuronales, Máquinas de Vectores Soporte, modelos de Markov, Clasificadores de K-vecinos cercanos y clasificadores Bayesianos han sido empleados para clasificación de intrones/exones con buenas precisiones de clasificación. Sin embargo, la mayoría de los métodos existentes son computacionalmente caros. En [57] los autores emplean Self Organizing Maps (SOMs) con el objetivo de encontrar las fronteras entre las secciones de ADN que codifican a proteínas y las que no. En [48] se estudian las características estadísticas asociadas para mejorar la precisión de clasificación, los autores emplean un clasificador de *k-vecinos cercanos,* las características resultantes son altamente discriminantes y según los autores produce resultados superiores en tareas de clasificación de exones/intrones. Zhang et al. emplean una SVM lineal, para ello emplean un kernel Bayesiano proyectando los datos de entrada dentro de un nuevo espacio de características [98]. Ogura et al. desarrollan una red neuronal empleando un algoritmo de propagación hacia atrás [62]. Los autores entrenan tres redes neuronales con arreglos de bases adyacentes a los lugares de empalme para mejorar la predicción de

los lugares de empalme.

En este Capítulo se presenta un nuevo algoritmo para detectar intrones y exones dentro de cadenas de ADN empleando Máquinas de Vectores Soporte y árboles de decisión para reducir el tiempo de entrenamiento de las SVM cuando el conjunto de datos es muy grande. Las SVM obtienen muy buenos resultados debido a su poder discriminativo, sin embargo éstas tienen problemas al trabajar en conjuntos de datos grandes, en este Capítulo se propone un algoritmo que enfrenta este problema sin sacrificar precisión. El algoritmo además de obtener buenos resultados en conjuntos de datos medianos está diseñado para trabajar con grandes conjuntos de datos, reduciendo el tiempo de entrenamiento empleado por las SVM. El Capítulo está organizado de la siguiente manera; en la Sección 1, se presenta una pequeña introducción a Biología y se define el problema y su importancia. En la Sección 2 se presenta el algoritmo de SVM basado en árboles de decisión. La Sección 3 se presenta el algoritmo de codificación del ADN y el empleo del algoritmo de clasificación en su totalidad. La Sección 4 presenta los resultados experimentales obtenidos con el método propuesto y finalmente en la Sección 5 se muestran las conclusiones del Capítulo.

7.1. Preliminares

7.1.1. El Ácido Desoxirribonucleico

El Ácido Desoxirribonucleico (ADN) es un ácido nucleico que contiene las instrucciones genéticas usadas en el desarrollo [52] [48]. El papel principal de las moléculas de ADN es el de ser portador y transmisor entre generaciones de información genética. Los segmentos de ADN que llevan está información genética se llaman genes. Las cuatro bases que se encuentran en el ADN son la adenina (A), citosina (C), guanina (G) y timina (T). Cada una de estas cuatro bases está unida al armazón de azúcar-fosfato a través del azúcar para formar el nucleótido completo (base-azúcar-fosfato). Se estima que el genoma humano tiene alrededor de 3000 millones de pares base. A la totalidad de

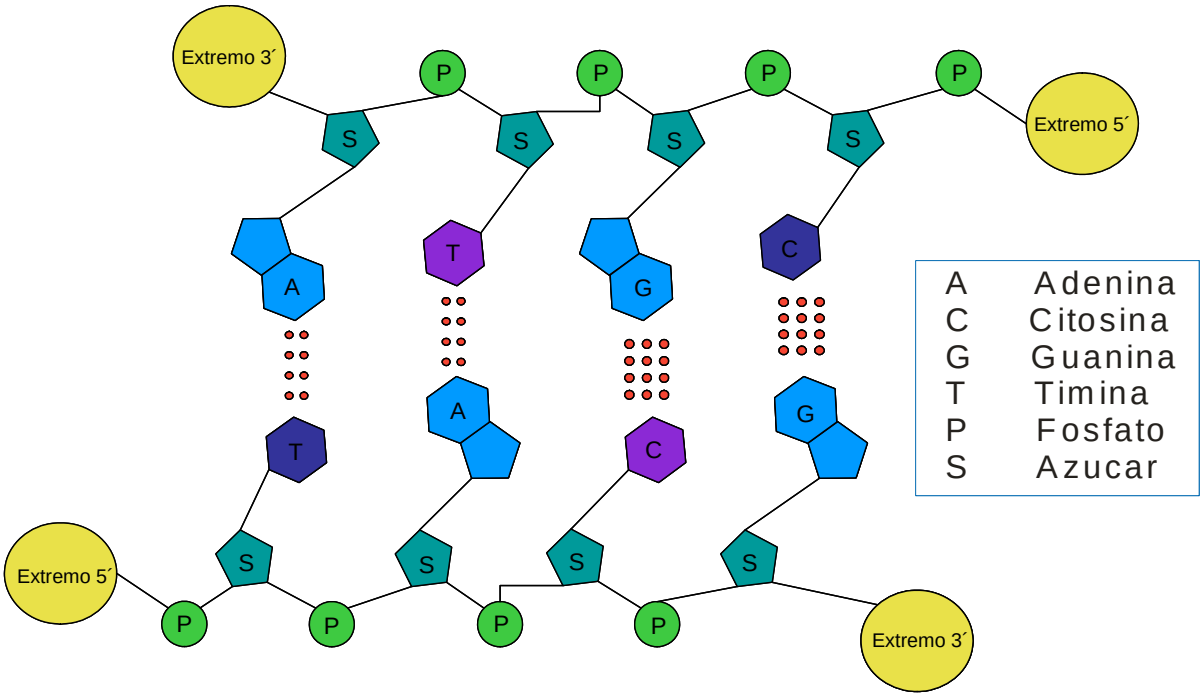

Figura 7.1: Ácido Desoxirribonucleico con las bases que la conforman

la cadena de ADN se le llama genoma de un organismo, la cadena en su totalidad es el conjunto de información que tiene como función contener toda la información necesaria para construir y sostener el organismo en el que reside, esta información es transmitida de generación en generación.

Los genes son codificados en pequeños fragmentos de ADN que están dispersados en el genoma, cada gene contiene información para producir una sola proteína, i.e., el ADN en el genoma de un organismo podría dividirse conceptualmente en dos, el que codifica las proteínas (los genes) y el que no codifica. Sólo una pequeña fracción del genoma codifica proteínas. Por ejemplo, sólo alrededor del 2 % del genoma humano codifica en proteínas, mientras que más del 90 % consiste de ADN no codificante.

El ADN no codificante corresponde a secuencias del genoma que no generan una proteína (procedentes de transposiciones, duplicaciones, translocaciones y recombinaciones

de virus, etc.), incluyendo los intrones. Corresponde a más del 90 % del genoma humano, que cuenta con 20.000 ó 25.000 genes.

Por otro lado, algunas secuencias de ADN desempeñan un papel estructural en los cromosomas: los telómeros y centrómeros contienen pocos o ningún gen codificante de proteínas, pero son importantes para estabilizar la estructura de los cromosomas. Algunos genes no codifican proteínas (la Figura 7.2 muestra los procesos de transcripción y Splicing para la obtención de proteínas), pero sí se transcriben en ARN: ARN ribosómico, ARN de transferencia, ARN de interferencia (ARNi, que son ARN que bloquean la expresión de genes específicos). La estructura de intrones y exones de algunos genes (como los de inmunoglobulinas y protocadherinas) son importantes por permitir cortes y empalmes alternativos del pre-ARN mensajero que hacen posible la síntesis de diferentes proteínas a partir de un mismo gen (sin esta capacidad no existiría el sistema inmune, por ejemplo). Algunas secuencias de ADN no codificante representan pseudogenes que tienen valor evolutivo ya que permiten la creación de nuevos genes con nuevas funciones. Otros ADN no codificantes proceden de la duplicación de pequeñas regiones del ADN; esto tiene mucha utilidad ya que el rastreo de estas secuencias repetitivas permite estudios sobre el linaje humano.

En un gen, la secuencia de nucleótidos a lo largo de una hebra de ADN se transcribe a un ARN mensajero (ARNm) y esta secuencia a su vez se traduce a una proteína que un organismo es capaz de sintetizar o "expresar" en uno o varios momentos de su vida, usando la información de dicha secuencia.

La relación entre la secuencia de nucleótidos y la secuencia de aminoácidos de la proteína viene determinada por el código genético, que se utiliza durante el proceso de traducción o síntesis de proteínas. La unidad codificadora del código genético es un grupo de tres nucleótidos (triplete), representado por las tres letras iniciales de las bases nitrogenadas (por ej., ACT, CAG, TTT). Los tripletes del ADN se transcriben en sus bases complementarias en el ARN mensajero, y en este caso los tripletes se denominan codones (para el ejemplo anterior, UGA, GUC, AAA). En el ribosoma cada codón del ARN mensajero interacciona con una molécula de ARN de transferencia (ARNt o tRNA)

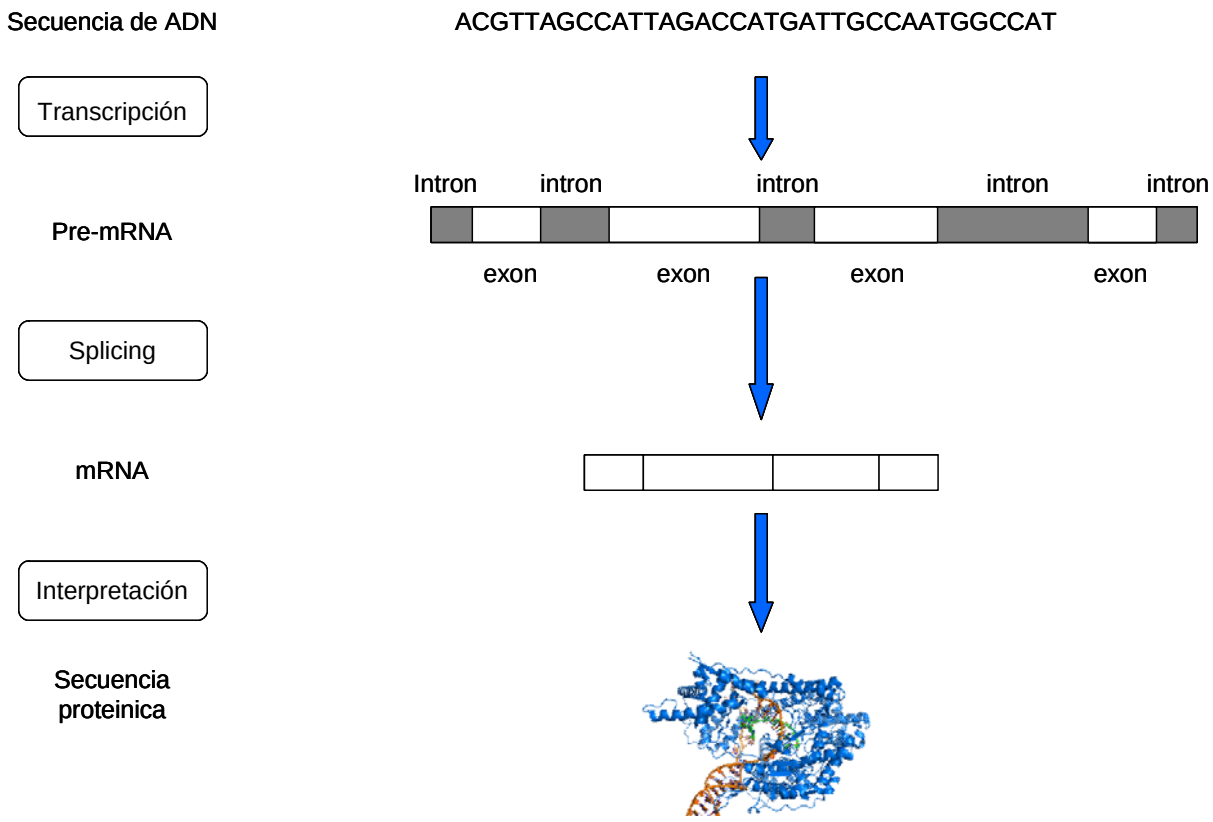

Figura 7.2: Transcripción del ADN en proteina

que contenga el triplete complementario, denominado anticodón. Cada ARNt porta el aminoácido correspondiente al codón de acuerdo con el código genético, de modo que el ribosoma va uniendo los aminoácidos para formar una nueva proteína de acuerdo con las "instrucciones" de la secuencia del ARNm. Existen 64 codones posibles, correspondiendo más de uno para cada aminoácido; algunos codones indican la terminación de la síntesis, el fin de la secuencia codificante; estos codones de terminación o codones de parada son UAA, UGA y UAG (en inglés, nonsense codons o stop codons).

En la actualidad los avances en obtención de secuencias de ADN han provocado que enormes cantidades de informacion sobre secuencias de ADN sean almacenadas en bases

de datos. Sin embargo, son necesarios métodos precisos para detectar los sitios de empalme en secuencias de ADN que enfrenten las necesidades actuales referentes a tamaño del conjunto de datos. La predicción precisa de sitios de empalme es un paso fundamental para el estudio de genes eucariotas. El problema de reconocimiento de genes puede ser catalogado como un problema de clasificación, en donde dado un conjunto de patrones de entrada que describan las diferentes clases existente, el objetivo es encontrar la mejor frontera de decisión, en este caso un conjunto de secuencias de DNA que caracterizen completamente una frontera de decisión para detectar exones. La mejor frontera de decisión será aquella que obtenga los mejores resultados al generalizar. En ese sentido, las SVM han sido empleadas en diversos campos de aplicación en los últimos años, la principal ventaja de estas, es su poder discriminativo y capacidad de generalización. Sin embargo, su principal desventaja es debido a su gran dependencia del conjunto de datos de entrada, es decir no están diseñadas para trabajar con grandes conjuntos de datos. En este Capítulo, se propone un algoritmo para clasificación de intrones y exones dentro de cadenas de ADN empleando SVM, el algoritmo es modificado con el objetivo de reducir el tiempo de entrenamiento de las SVM cuando el conjunto de datos es muy grande.

7.2. Algoritmo Bio-SVM^2

En esta Sección se describe el algoritmo propuesto empleando las SVM propuestas en la Sección anterior. Sin embargo, un paso muy importante en la predicción de exones es el encriptamiento de las secuencias de ADN. El encriptamiento está basado en recuperar las características más importantes dentro de una secuencia de ADN.

En la literatura actual, existen varias formas de representar mediante vectores las características de las secuencias de ADN [98][37]. A este procedimiento se le llama *encriptar el ADN*. En la literatura, se emplean los diversos métodos existentes basándose en las características que creen más importantes para representar la secuencia estudiada. El encriptamiento esparcido es un esquema de encriptamiento ampliamente usado, este esquema representa cada base con 4 bits: $A \rightarrow 1000, C \rightarrow 0100, G \rightarrow 0010$ y

$T \rightarrow 0001$ [37]. Suponiendo que se tiene la secuencia de ADN siguiente ACTTCGTAAGAT. Empleando encriptamiento esparcido. La secuencia anterior sería representada como: 1000 | 0100 | 0001 | 0001 | 0100 | 0010 | 0001 | 1000 | 1000 | 0010 | 1000 | 0001. Donde | es únicamente un separador virtual usado para ilustrar el ejemplo.

Algunos autores argumentan algunas desventajas asociadas a este método de encriptado, como por ejemplo que la secuencia de datos es linealmente no separable para el caso de separar falsos y verdaderos sitios de empalme []. Sin embargo, es posible emplear algunos *kernels* con el propósito de enfrentar este problema. Aunque emplear *kernels* incrementa la complejidad computacional, sobre todo cuando se trabaja con grandes conjuntos de datos, el método empleado aquí para reducir la complejidad computacional reduce en gran manera este problema.

Con el propósito de mejorar la precisión de clasificación al predecir los sitios de empalme con SVM, en este ejemplo se emplearon 20 características adicionales con el método de encriptamiento esparcido. Los primeros 16 componentes definen los pares de bases en las secuencias, que son descritos mediante $\beta = \{(x_{AA}),\ (x_{AC}),\ (x_{AG}),\ (x_{AT}),(x_{CA}),\ (x_{CC}),(x_{CG}),(x_{CT}),(x_{GA}),(x_{GC}),(x_{GG}),(x_{GT}),(x_{TA}),(x_{TC}),(x_{TG}),(x_{TT})\}$. Cuando algún par se encuentra dentro de la secuencia a encriptar su presencia es marcada con 1 mientras que la ausencia de este dentro de la secuencia es marcada con 0. La secuencia de ADN, ACTTCGTAAGAT puede ser encriptada mediante este método de la siguiente manera: 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1. Los siguientes 4 componentes que se emplearon corresponden a la profusión de cada nucleótido en la cadena, es decir, cada secuencia es representada mediante un vector con 4 características que representan la presencia de relativa de cada base dentro de la secuencia de ADN. La secuencia ACTTCGTAAGAT puede ser encriptada mediante este método de la siguiente manera: $\gamma = \{f_A, f_C, f_G, f_T\} = \{0{,}33, 0{,}16, 0{,}16, 0{,}33\}$.

El método de encriptamiento propuesto permite obtener los nucleótidos que la conforman en su totalidad, codificar los pares de nucleótidos reflejaría de alguna forma la importancia de algunos pares sobre otros, además de obtener la profusión de cada nucleótido dentro de la secuencia de ADN. La cadena completa codificada la conforman

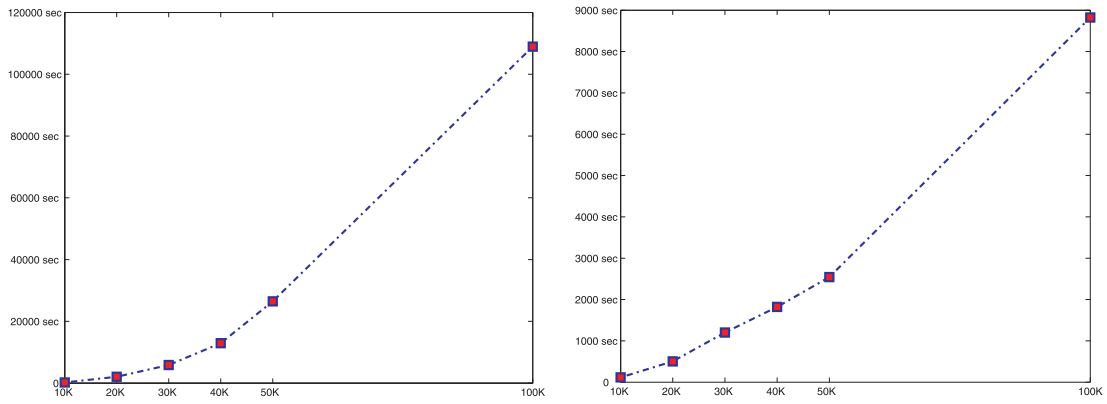

Figura 7.3: Tiempo de entrenamiento de SMO y LibSVM

características continuas como binarias. La secuencia de ADN ejemplificada anteriormente al ser encriptada con el método propuesto quedaría de la siguiente forma: 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1 |1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1 | 0.33, 0.16, 0.16, 0.33. Donde | es un separador virtual que tiene como único objetivo mostrar los puntos de unión entre los tres métodos de encriptamiento empleados. Una vez descrito el método de encriptamiento empleado, la SVM puede interpretar los valores y discriminar entre las distintas clases dadas. La Figura 7.4 describe el algoritmo empleado en su totalidad.

Un proceso de clasificación incluye una fase de entrenamiento y una fase de prueba. En la fase de entrenamiento, el conjunto de entrenamiento es usado para decidir como los parámetros deben ser pesados y separados con el objetivo de encontrar una función de decisión que separe los diferentes tipos de objetos. Esta fase intenta descubrir una representación óptima de un conjunto de datos con funciones de membresía conocidas. Por otro lado en la fase de prueba, los pesos determinados en la fase de entrenamiento son aplicados a un conjunto de objetos con etiquetas de clase desconocidas con el objetivo de determinar sus clases.

Durante la fase de entrenamiento las SVM encuentran un hiperplano de clasificación óptimo que divide las clases. En el caso de que los puntos de entrada no puedan ser separados mediante un hiperplano, las SVM mapean los datos de entrada a un espacio de características altamente dimensional donde éstos puedan ser separados mediante un hiperplano. Sin embargo, encontrar este hiperplano de clasificación es muy costoso computacionalmente hablando. Ya que resolver un problema de programación cuadrática tiene una complejidad cuadrática, las SVM necesitan enormes cantidades de tiempo y memoria computacional. La Figura 7.3 muestra cuán inescalable puede llegar a ser entrenar una SVM con grandes conjuntos de datos. Considerando el caso más simple de clasificación, un conjunto de datos con dos clases en dos dimensiones es posible ver cuán costoso puede llegar a ser entrenar un conjunto de datos con SVM. La Figura 7.3 muestra los resultados sobre el conjunto de datos Checkerboard empleando las implementaciones de SVM para grandes conjuntos de datos; SMO -a)- y Libsvm -b)- con un conjunto de datos con 100,000 registros.

Por otro lado, en la fase de prueba las SVM únicamente determinan la clase de cada vector de entrada a partir de los pesos obtenidos durante la fase de entrenamiento. Los vectores soporte presentan las características necesarias para obtener un hiperplano discriminante. Es decir, estos contribuyen directamente para encontrar el hiperplano óptimo. Por el contrario, aquellos que se encuentran alejados del hiperplano de clasificación no contribuyen en forma alguna en su obtención. La complejidad de la fase de prueba es dependiente del conjunto de vectores soporte. Dado que el conjunto de vectores soporte es una pequeña parte del conjunto entero de entrenamiento, el tiempo de prueba debiera ser muy pequeño. Sin embargo, en algunos casos, el número de vectores soporte es muy grande y por lo tanto el tiempo de prueba podría llegar a ser muy grande.

7.2.1. Árboles de Decisión

Las técnicas de árboles de decisión han llegado a ser en los últimos años una de las herramientas de clasificación y regresión más populares. La principal ventaja de

los árboles de decisión es debido al hecho que, en contraste a la redes neuronales, los árboles de decisión representan reglas. Reglas que pueden ser fácilmente expresadas de tal forma que cualquier persona pueda entenderlas. Regularmente un algoritmo de árboles de decisión inicia con el conjunto de datos entero, divide los datos dentro de dos o más subconjuntos basándose en los valores de los atributos y repetidamente divide cada suconjunto dentro de subconjuntos más finos hasta que el tamaño del árbol alcanza un nivel apropiado.

El proceso de modelado total puede ser representado mediante una estructura de árbol y el modelo generado puede ser sintetizado como un conjunto de reglas "Si-Entonces". Los árboles de decisión son fáciles de interpretar, computacionalmente económicos y capaces de afrontar conjuntos de datos con ruido. En este algoritmo se utilizó el algoritmo C4.5 [71] para construir los árboles de decisión. La selección del mejor atributo esta basada en la ganacia de radio Gain (S,A) donde S es el conjunto de registros y A es un atributo. Esta ganancia define la reducción esperada en entropía debido al ordenamiento de A. Está es calculada como

$$Gain(S, A) = Entropia(S) - \sum_{v \in Valor(A)} \frac{|S_v|}{|S|} Entropia(S_v) \tag{7.1}$$

En general, si se tiene una distribución de probabilidad $P = (p_1, p_2, \ldots, p_n)$ entonces la información dada por esta distribución es llamada entropía de P y está dada por:

$$Entropia(P) = \sum_{i=1}^{n} p_i \log_2 p_i \tag{7.2}$$

Una forma de reducir el tiempo de entrenamiento es eliminando aquellos datos alejados del hiperplano de clasificación, mientras que los datos cercanos al hiperplano son obtenidos y guardados. Sin embargo no es posible saber dónde está el hiperplano hasta que utilizamos SVM sobre un conjunto de datos. Resulta claro que podemos emplear SVM sobre un conjunto muy pequeño de datos y emplear el hiperplano obtenido para eliminar datos alejados y obtener aquellos cercanos al hiperplano del conjunto total de datos.

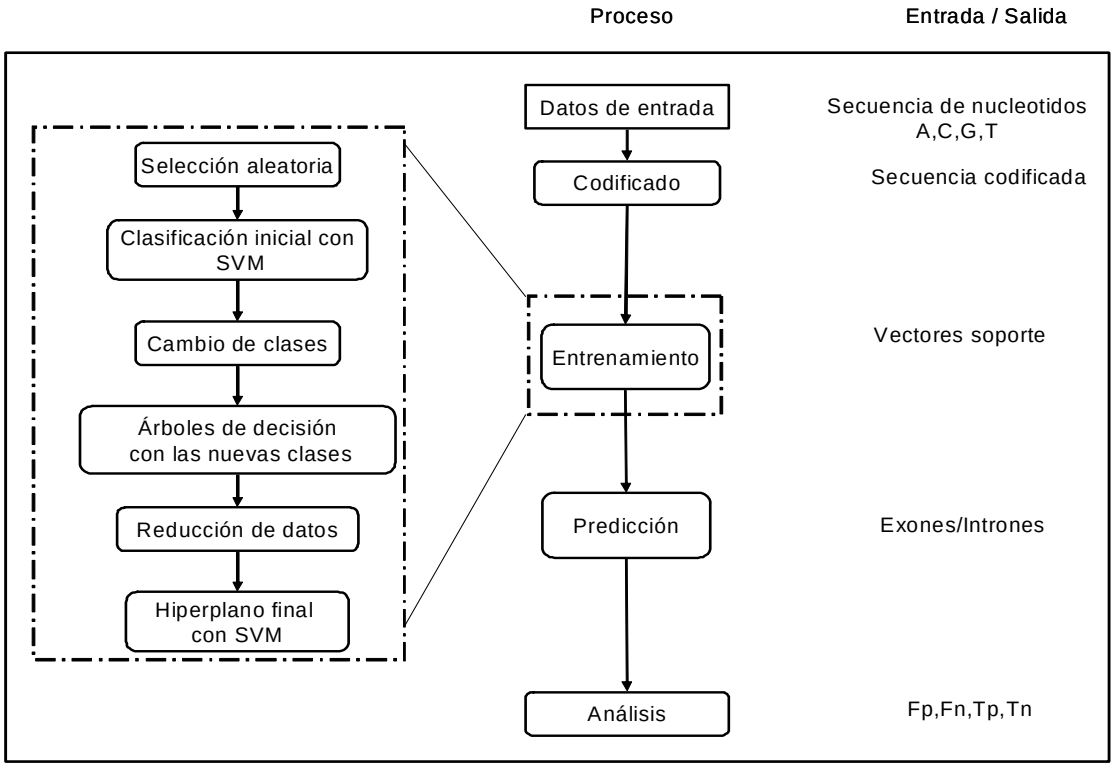

Figura 7.4: Algoritmo de clasificación basado en arboles de decisión

De acuerdo al análisis anterior, es posible realizar una modificación al algoritmo de entrenamiento de las SVM. La Figura 7.4 muestra el enfoque propuesto en 6 pasos 1) Selección aleatoria de un conjunto de datos muy pequeño 2) Una primera etapa de SVM sobre este conjunto de datos, 3) Obtención de los vectores soporte y cambio de clases, 4) Nuevo hiperplano empleando árboles de decisión, 5) Prueba del árbol obtenido sobre el conjunto de datos original para obtener el conjunto de datos reducido y 6) Etapa final de clasificación con SVM.

1) Selección de un conjunto inicial

Con el objetivo de obtener un pequeño porcentaje de elementos del conjunto original, se utiliza un algoritmo de selección aleatoria, El algoritmo escoge un conjunto inicial dependiendo de los tamaños de las clases. Asumiendo que (X, Y) es el conjunto de patrones de entrenamiento, donde $X = \{x_1, x_2, ..., x_n\}$ es el conjunto de datos de entrada, x_i puede ser representado por un vector de dimensión p, esto es, $x_i = (x_{i_1}...x_{ip})^T$, $Y = \{y_1, y_2, \ldots, y_n\}$ es el conjunto de datos con sus etiquetas, donde la etiqueta $y_i \in \{-1, 1\}$. El objetivo es seleccionar un conjunto de datos inicial pequeño $C = \{x_i, y_i\}_{i=1}^{l}$, $l \ll n$ a partir de X . Después de la selección aleatoria, se denota el conjunto de datos como C. Es claro que el conjunto de datos reducido C puede ser dividido en dos subconjuntos, uno que contiene etiquetas positivas y otro conteniendo etiquetas negativas, esto es,

$$\begin{aligned} C^+ &= \{\cup C_i \mid y = +1\} \\ C^- &= \{\cup C_i \mid y = -1\} \end{aligned} \tag{7.3}$$

2) La primera fase de SVM

La Fig. 7.5 a) muestra el conjunto de datos original, es fácil identificar los datos con etiquetas negativas (-) y los datos con etiquetas positivas (+). Después de la selección aleatoria, se obtiene un conjunto reducido de datos como se muestra en la Fig. 7.5 b). De la ecuación 7.3, se tiene que, C^-consiste de aquellos datos que están representados por (-) Figura 7.5 b), mientras que C^+consiste de aquellos datos que están representados por (+)

en la Figura 7.5 b). Únicamente estos datos serán usados como datos de entrenamiento en la primera etapa de clasificación con SVM. Una vez que se ha seleccionado un conjunto de datos inicial, se clasifican los datos empleando una primera fase de SVM, el hiperplano obtenido es únicamente un bosquejo de la forma en que están distribuidas las clases, lo que nos servirá en los posteriores pasos para reducir el conjunto total de datos. Después de la primera fase de clasificación con SVM el hiperplano obtenido está dado por:

$$\sum_{k \in V_1} y_k \alpha^*_{1,k} K\left(x_k, x\right) + b^*_1 = 0 \tag{7.4}$$

donde V_1 representa el conjunto de datos que son vectores soporte en la primera fase. Aquí, el conjunto de datos de entrenamiento está dado por $C^+ \cup C^-$ obtenido mediante selección aleatoria. La Fig. 7.5 b) y c) ilustra el pequeño conjunto de datos de entrenamiento, el hiperplano obtenido y los vectores soporte en la primera fase de entrenamiento con SVM.

3) Cambio de clases

En el paso anterior, se obtuvo un hiperplano de clasificación. Sin embargo, tanto el hiperplano como los vectores soporte obtenidos no son lo suficientemente representativos. No obstante, el hiperplano obtenido no da una referencia acerca de la distribución de los datos. Del paso anterior, se obtienen un conjunto de vectores soporte y datos que no son vectores soporte. Los datos que son vectores constituyen los datos más importantes en el conjunto de datos pequeño, a éstos se les asigna una etiqueta extra E^{+1}, los datos que no son vectores soporte representan aquellos que no son importantes o son innecesarios por estar alejados del hiperplano de clasificación, a éstos se les asigna la etiqueta extra E^{-1}.

Formalmente el conjunto de datos inicial pequeño está dado por $C = \{x_i, y_i\}_{i=1}^{l}$ donde $l \ll n$, Los vectores soporte en la primera fase de SVM están dados por

$$\{(x^*_1, y_1), (x^*_2, y_2), \ldots, (x^*_k, y_k)\}, 1, 2, \ldots, k \in V_1$$

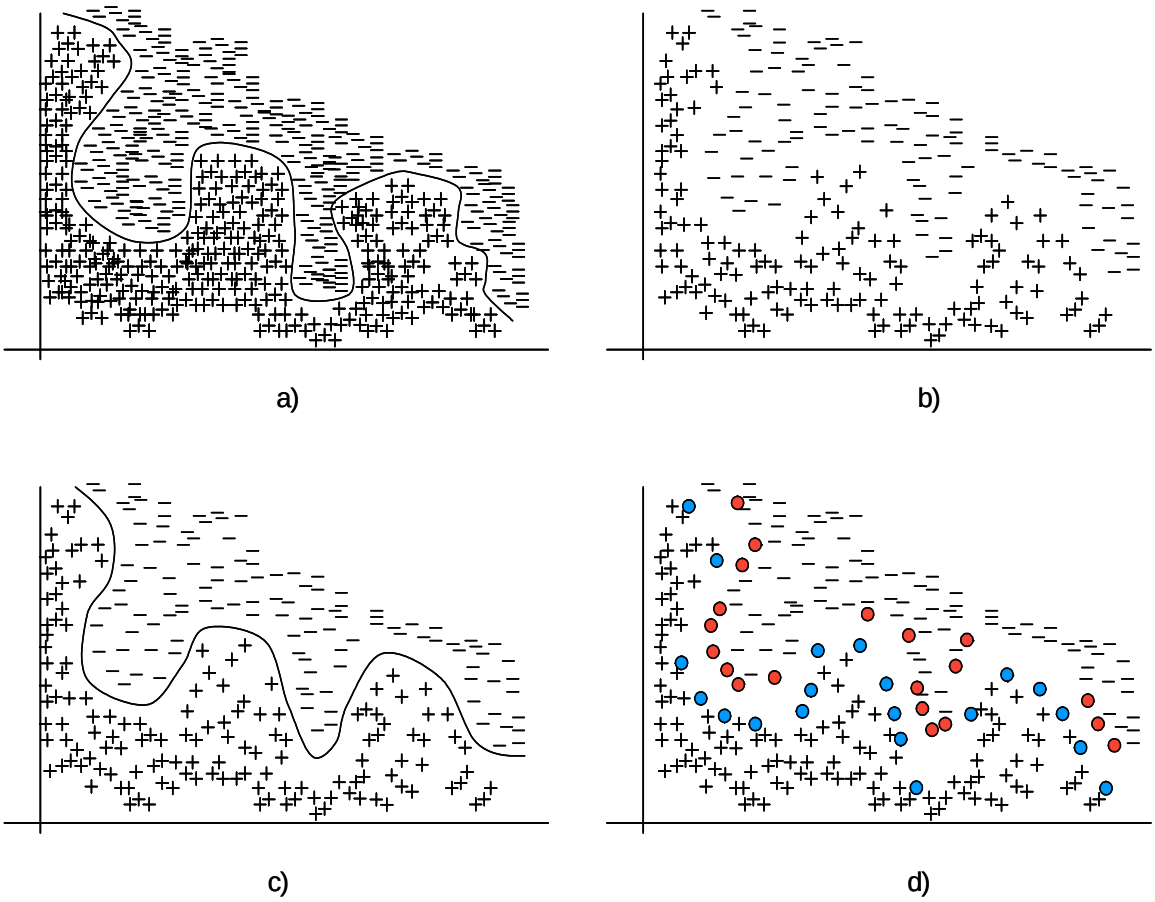

Figura 7.5: Etapas iniciales del algoritmo de clasificación propuesto

mientras que el conjunto de datos que no son vectores soporte está dado por

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}, 1, 2, \ldots, m \notin V_1$$

donde V_1 representan los vectores soporte en la primera fase, es claro que $\{x_i, y_i\}_{i=1}^{l} = \{x_k, y_k\} \cup \{x_m, y_m\}$ de aquí se tiene que el conjunto $\{x_i, y_i\}_{i=1}^{k} \in V_1$ y su nueva clase es $\{x_i, y_i\}_{i=1}^{k} \in E^{+1}$ mientras que para $\{x_i, y_i\}_{i=1}^{m} \notin V_1$ su nueva clase es E^{-1}.

4) Entrenamiento del árbol de decisión

Con el objetivo de obtener todos los puntos que están cercanos al hiperplano de clasificación del conjunto de datos original, en esta fase se entrena un árbol de decisión con los datos con las nuevas clases. El árbol de decisión es entrenado con la nueva distribución de datos, la cual está dada por:

$$(\mathbf{x}_{a1}, \mathbf{y}_{a1}), (\mathbf{x}_{a2}, \mathbf{y}_{a2}), \ldots, (\mathbf{x}_{an}, \mathbf{y}_{an}) \tag{7.5}$$

i.e. $X = \{x_{ai}, y_{ai}\}_{i=1}^{q}$ donde $q = k + m$, $x_{ai} \in R^d$ y $y_{ai} \in (+1, -1)$, donde todos los datos con etiqueta $y_{ai} \in (+1)$ representan los vectores soporte obtenidos en la etapa previa y todos los puntos con etiqueta $y_{ai} \in (-1)$ representan los datos que no son vectores soporte.

Con el propósito de encontrar una función que divida el conjunto de datos cercanos al hiperplano de los alejados al hiperplano se entrena el árbol de decisión, de donde se obtiene una función de decisión que separa los ejemplos positivos de la nueva distribución (los datos que están cerca a los vectores soporte) de los ejemplos negativos (los datos que están alejados de los vectores soporte). Finalmente la función de decisión de la nueva distribución está dada por el árbol de decisión. Es claro que es necesario un criterio de paro con el fin de obtener un buen clasificador. Existen varias formas para asignar un criterio de paro. Por ejemplo, es posible parar el algoritmo cuando se ha alcanzado cierto tamaño el árbol o un cierto número de nodos o cuando cierta precisión de clasificación es alcanzada. En este caso, el criterio de paro empleado es el siguiente: si la precisión sobre el conjunto de datos de entrenamiento excede el 85 % el entrenamiento del árbol llega a

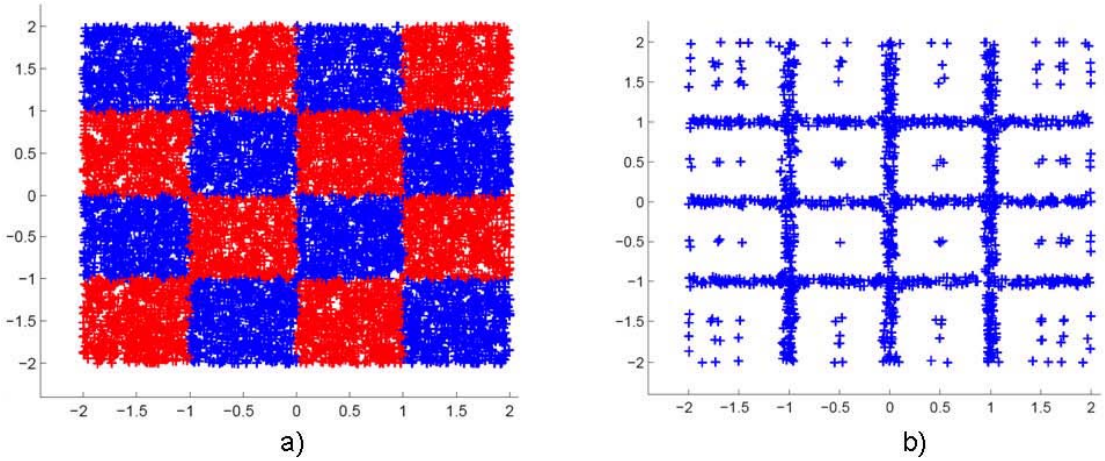

Figura 7.6: Conjunto de datos de entrada y VS de la primera fase de SVM.

su fin. En este caso, no es necesario obtener un clasificador con un 100 % de precisión, ya que al probarlo con los datos originales nos daría únicamente los datos más cercanos al hiperplano. Esto podría reducir al máximo el conjunto de datos final, por lo tanto se necesita cierta imprecisión en esta fase con el objetivo de mejorar el hiperplano de clasificación.

5) Recuperación de datos importantes

Es claro que es posible obtener los datos cercanos a los vectores soporte a partir del conjunto de datos original. En la primera fase de clasificación con SVM, el conjunto de datos es únicamente un pequeño porcentaje del conjunto de datos original. Por lo tanto, el hiperplano de decisión obtenido no puede ser lo suficientemente preciso, a pesar de ello, éste nos da una clara referencia acerca de los datos que pueden ser eliminados. No obstante, es entendible que muchos datos útiles, que podrían mejorar la precisión de clasificación no han sido seleccionados durante la etapa de selección aleatoria. Por lo tanto, una idea natural sería recuperar los datos localizados entre los vectores soporte con diferente clase y aquellos que se encuentran cercanos a estos y emplear todos estos datos

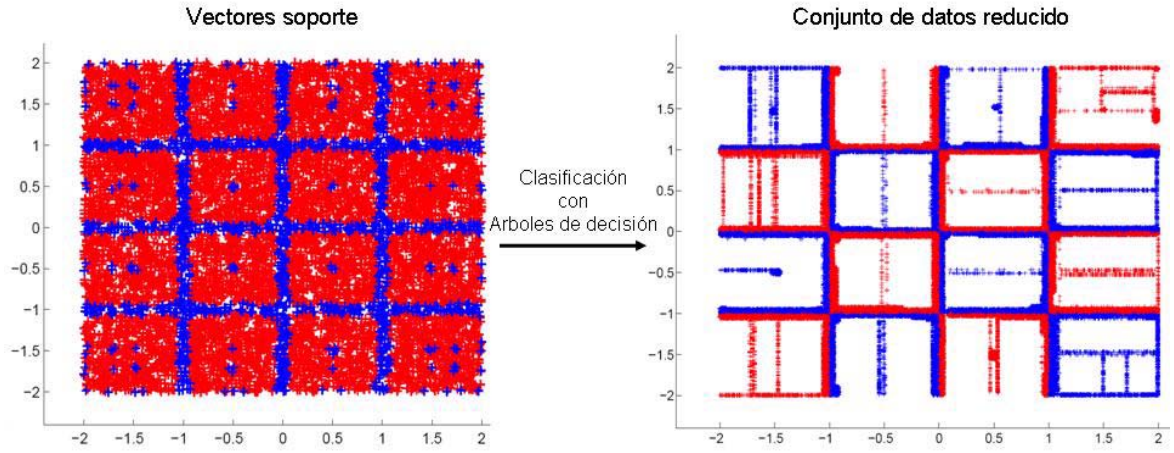

Figura 7.7: VS y conjunto de datos reducido empleando el algoritmo propuesto.

recuperados para refinar el hiperplano empleando SVM una vez más. Con el objetivo de refinar el primer hiperplano de clasificación obtenido, el árbol de decisión obtiene los puntos cercanos a los vectores soporte. La Fig.7.8 a), b) y c) muestra los pasos empleados por el algoritmo para recuperar los más importantes datos. El conjunto de datos recuperado está dado por

$$\{x_{pi}, y_{pi}\} \cup (\{x_i, y_i\}_{i=1}^{k} \in V_1),$$

donde x_{pi} representa todos los puntos cercanos al hiperplano de decisión obtenido en la primera fase, los cuales poseen una etiqueta positiva, mientras que V_1 representa los vectores soporte de la primera fase.

6) Etapa final de clasificación con SVM

De los datos recuperados se obtiene un nuevo conjunto de datos de entrenamiento, En esta etapa, se emplea una vez más SVM con el objetivo de obtener el hiperplano

final de clasificación que es dado por

$$\sum_{k \in V_2} y_k \alpha^*_{2,k} K\left(x_k, x\right) + b^*_2 = 0 \tag{7.6}$$

donde V_2 es conjunto de vectores soporte encontrado en esta fase final. Generalmente, el hiperplano (7.6) es cercano al hiperplano (7.4). No obtante que el hiperplano final tiende a ser más preciso.

Es claro que los datos recuperados consisten de los datos más útiles desde el punto de vista de optimización, ya que los datos más alejados del hiperplano de decisión han sido eliminados. Por otro lado, la mayoría de los datos cercanos al hiperplano real han sido incluidos por la técnica de recuperación de datos, el hiperplano (7.6) es pues una aproximación del hiperplano obtenido con una SVM empleando SMO con todos los datos originales. La Fig.7.8 d) ilustra los datos de entrenamiento y el hiperplano en la segunda fase de clasificación.

7.3. Resultados Experimentales

Ejemplo 7.1 *El conjunto de datos empleado para este estudio es la secuencia de genes de empalme en primates tomada de Genbank 64.1 (sitio ftp: genbank.bio.net). El conjunto de datos de ADN contiene 3190 secuencias de ADN con 62 nucleótidos cada secuencia. El conjunto de datos contiene 767 cotas de exon/intron (referidas como sitios EI), 768 cotas intron/exon (referidas como sitios IE) y 1655 secuencias de ninguno de los dos.*

El método propuesto es aplicado para predecir los datos. En los experimentos se utilizó el kernel función radial base (RBF), la cual es definida como

$$K(x_i - x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \gamma > 0$$

donde γ es uno de los parámetros empleados con las SVM, mientras que el otro es C. Con el objetivo de encontrar los mejores parámetros se utilizó un método de búsqueda de malla. Los resultados calculados con el método propuesto son mostrados en las Tablas

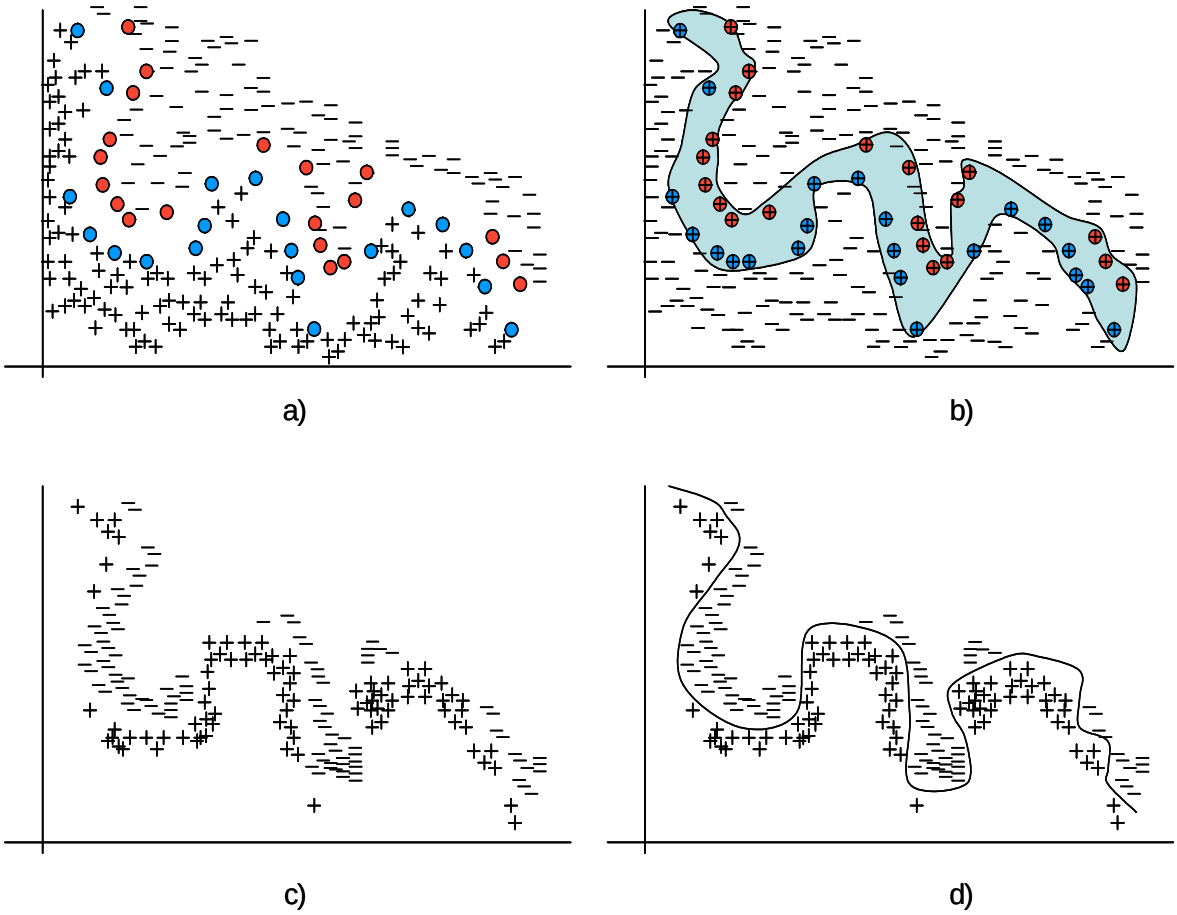

Figura 7.8: Etapas finales del algoritmo de clasificación propuesto

7.1 y 7.2, T_p representa el número nucleótidos que codifican en proteínas y que han sido correctamente clasificados (positivos acertados), T_N representa a aquellos nucleótidos que no codifican en proteínas y que han sido correctamente clasificados como secuencias no codificantes (negativos acertados). F_P representa a aquellos nucleótidos que codifican en proteínas y que no han sido correctamente clasificados (Positivos no acertados), mientras que F_N representa a aquellos nucleótidos que no codifican en proteínas y que no han sido correctamente clasificados (Negativos no acertados). S_n^{true} es la proporción entre el número de sitios de empalme positivos correctamente clasificados y el número total de sitios de empalme en el conjunto de datos de prueba, i.e.,

$$S_n^{true} = \frac{T_p}{T_p + F_N}$$

El desempeño de la SVM propuesta para calcular los sitios de empalme negativos correctamente clasificados también es evaluado mediante S_n^{false} la cual es calculada mediante

$$S_n^{false} = \frac{T_N}{T_N + F_P}$$

S_n es la proporción de los sitios candidatos en el conjunto de datos de prueba que son clasificados correctamente y está dada por

$$S_n = \frac{N_c}{N_t}$$

donde N_c es el número de exones correctamente clasificados en el conjunto de prueba y N_t es el número de total de sitios exones en el conjunto de prueba.

En los experimentos realizados se utilizó 80 % de los datos para entrenar la SVM y 20 % para prueba. La SVM fue entrenada y evaluada 20 veces, en cada evaluación se determinó el conjunto de datos de entrenamiento y el de prueba de forma aleatoria. Los resultados de cada evaluación son mostrados en la Tabla 7.1 y la Tabla 7.2. La Tabla 7.1 muestra los resultados de cada evaluación, el promedio de precisión de todas las evaluaciones (Acc_Av) y el promedio de la desviación estándar (Av_SD). La Tabla 7.2

Tabla 7.1: Resultados de 20 corridas realizadas con el algoritmo Bio-SVM^2

	TrD	TeD	Acc	S_n^{true}	S_n^{false}
	2552	638	97.962382	0.983278	0.976401
	2552	638	98.119122	0.981366	0.981013
	2552	638	97.962382	0.980263	0.979042
	2552	638	98.589342	0.981707	0.990323
	2552	638	98.589342	0.990566	0.981250
	2552	638	97.805643	0.963190	0.993590
	2552	638	98.432602	0.984127	0.984520
	2552	638	99.376301	0.996485	0.993937
	2552	638	98.119122	0.993031	0.971510
	2552	638	98.432602	0.987097	0.981707
	2552	638	98.432602	0.980263	0.988024
	2552	638	98.432602	0.990260	0.978788
	2552	638	97.962382	0.973941	0.984894
	2552	638	97.962382	0.984663	0.974359
	2552	638	98.746082	0.983607	0.990991
	2552	638	99.216301	0.996429	0.988827
	2552	638	97.962382	0.980519	0.978788
	2552	638	97.962382	0.977636	0.981538
	2552	638	97.648903	0.963684	0.971261
	2552	638	97.805643	0.979381	0.976945
Acc_Av			98.268	0.982975	0.982585
SD			21.870007	0.218996	0.218452

TrD datos de entrenamiento, TeD datos de prueba, Acc precisión

Acc_Av precisión promedio, SD desviación estándar

Tabla 7.2: Resultados de precisión de clasificación del conjunto de datos Genbank 64.1

Genbank 64.1				
	Acc	S_n^{true}	S_n^{false}	S_n
Bp	99.37	0.996	0.993	0.997
Pp	97.64	0.963	0.971	0.974
Av	98.268	0.982	0.982	0.984
SD	21.87	0.2189	0.2184	0.217

muestra la evaluación con mejor precisión (Bp), con peor precisión (Pp), la media (Av) y la desviación estándar (SD) de los resultados obtenidos.

Ejemplo 7.2 *El segundo ejemplo usado es el conjunto de datos Acceptor/Donor, obtenido de http://www2.fml.tuebingen.mpg.de/raetsch/projects/. El conjunto de datos Acceptor contiene 91546 datos de entrenamiento y 75905 datos de prueba (2132 puntos positivos), mientras que el conjunto de datos Donnors contiene 89163 datos de entrenamiento y 73784 datos de prueba (2132 puntos positivos). En estos ejemplos, mostramos la diferencia en tiempo de entrenamiento entre el enfoque propuesto y otras implementaciones de SVM.*

En los experimentos realizados se utilizó la función kernel RBF. Con el objetivo de seleccionar los parámetros óptimos para la SVM tratamos con diferentes valores de C y γ empleando un método de búsqueda de malla. Para el conjunto de datos Acceptors, encontramos que el valor óptimo para γ es $\gamma = 0{,}027$, mientras que para el conjunto de datos Donors, el valor óptimo encontrado es $\gamma = 0{,}45$.

La Figura 7.9, muestra los tiempos de entrenamiento de algunas implementaciones de SVM como son Simple SVM y LibSVM en comparación con los tiempos de entrenamiento obtenidos con el enfoque propuesto. El tiempo de entrenamiento de Simple SVM es muy

Tabla 7.3: Resultados de precisión de clasificación del conjunto de datos Acceptor

Conjunto de datos Acceptor				
	EP		LIBSVM	
#	t	Acc	t	Acc
50000	8	95.9	2279	96.3
91546	72	98.2	6371	98.4

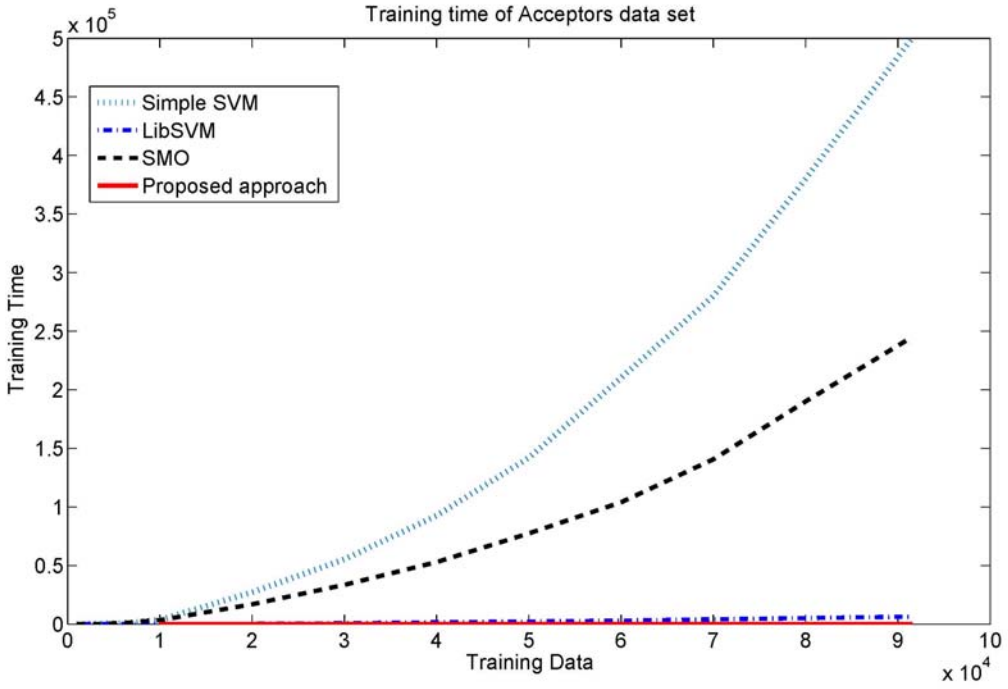

Figura 7.9: Tiempos de entrenamiento con diferentes implementaciones de SVM para el conjunto de datos Acceptors.

rápido en conjuntos de datos con dos dimensiones, aún cuando el conjunto de datos sea muy grande (como se pudo ver en el Capítulo 4). Cuando el número de dimensiones asociado al conjunto de datos de entrada crece, el tiempo de entrenamiento con Simple-SVM llega a ser mayor al empleado por LibSVM. En este caso, entrenar el conjunto de datos entero (91546 datos) donde cada dato es representado mediante 244 características, el tiempo de entrenamiento llega a ser demasiado costoso. La Figura 7.9 muestra que el tiempo de entrenamiento llega a ser alrededor de 500,000 segundos o 138 horas. SMO es una implementación de SVM muy rápida. Sin embargo, en este caso entrenar el conjunto de datos entero toma alrededor de 245,000 segundos o 68 horas. La más competitiva de éstas implemetaciones de SVM es LibSVM, ya que el tiempo de entrenamiento llevado a cabo con LibSVM es muy pequeño (6371 segundos) en comparación con Simple SVM y SMO. Sin embargo, el tiempo de entrenamiento del enfoque propuesto es aún menor (72 segundos), la Tabla 7.3 muestra los tiempos de entrenamiento y precisiones obtenidas con el enfoque propuesto y LibSVM.

Por otro lado, la Figura 7.10 muestra los tiempos de entrenamiento obtenidos con el conjunto de datos Donor, es claro que los tiempos de entrenamiento son muy similares a los obtenidos con el conjunto de datos Acceptor, debido a que son muy similares en tamaño e idénticos en el número de características asociadas.

La Tabla 7.4 muestra los resultados obtenidos de S_n^{true}, S_n^{false} y el error general. En la Tabla SSVM se refiere a la implementación Simple SVM, mientras que EP se refiere al enfoque propuesto. Los valores obtenidos de S_n^{true} con el enfoque propuesto son mejores que los obtenidos con GenScan cuyos resultados fueron reportados en [72].

7.4. Conclusiones

En este Capítulo se presentó un enfoque de clasificación con SVM para grandes conjuntos de datos empleando árboles de decisión para reducir de datos innecesarios. El enfoque propuesto supera la desventaja que supone entrenar grandes conjuntos de datos con kernels no lineales en datos biológicos. Ya que el enfoque propuesto reduce

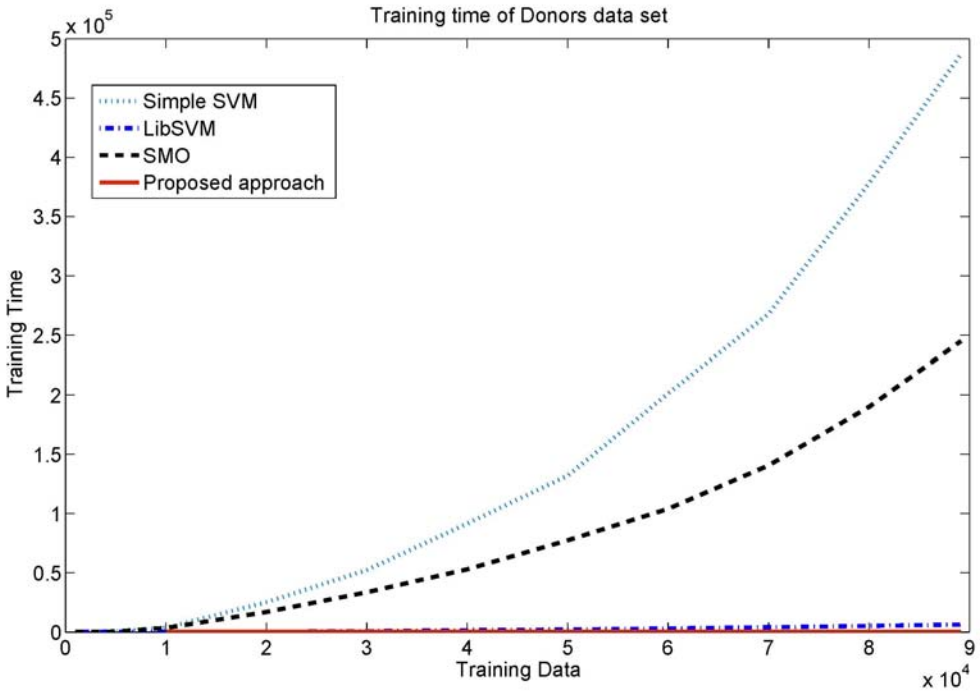

Figura 7.10: Tiempos de entrenamiento con diferentes implementaciones de SVM para el conjunto de datos Donnors.

Tabla 7.4: Resultados de clasificación del conjunto de datos Donor y Acceptor

	Acceptor			Donor		
	Error	S_n^{true}	S_n^{false}	Error	S_n^{true}	S_n^{false}
SMO	1.6	0.84	0.98	1.8	0.82	0.98
SSVM	2.4	0.72	0.97	2.6	0.71	0.97
Libsvm	1.6	0.84	0.98	1.8	0.82	0.98
EP	1.8	0.837	0.98	1.9	0.81	0.98

significativamente el conjunto de entrenamiento, eliminando aquellos datos menos significativos y conservando aquellos con más posibilidades de influir en el conjunto de entrenamiento. Experimentos realizados sobre conjuntos de datos biológicos muestran que el enfoque propuesto tiene una gran ventaja sobre conjuntos de datos grandes. Además el encriptamiento propuesto genera buenas precisiones de clasificación. Es claro que algunas características influyen de manera directa sobre la predicción de clasificación, un estudio sobre ello deberá realizarse con el objetivo de reducir el número de características asociadas a cada secuencia de nucleótidos.

Capítulo 8

Conclusiones

En esta Sección sintetizamos los resultados obtenidos, discutimos estos desde una amplia perspectiva y finalmente mostramos las futuras direcciones de investigación.

8.1. Conclusiones

Las SVM son un reciente método de clasificación con fundamentos teóricos y su buena capacidad de generalización, las SVM han llegado a ser en los últimos años uno de los métodos de clasificación más utilizado. Sin embargo, las SVM necesitan resolver un problema de programación cuadrática, requiriendo grandes cantidades de tiempo computacional y memoria en conjuntos de datos muy grandes. En este trabajo se proponen algoritmos basados en técnicas de agrupamiento y selección de datos para reducir el tamaño inicial del conjunto de datos, sin afectar significativamente la precisión de clasificación.

Para ello, se ha comparado el comportamiento de técnicas de agrupamiento. Además, se ha estudiado el problema de escalabilidad que presentan los algoritmos de agrupamiento, proponiéndose una versión del algoritmo de esferas de cerradura mínima para abordar este problema y utilizando selección aleatoria en otro caso. A continuación se resumen brevemente los resultados obtenidos y presentan algunas conclusiones sobre los mismos.

1. El algoritmo FCM-SVM^2, fue diseñado para trabajar en conjuntos de datos de mediano tamaño, obteniendo los siguientes resultados:

   - La obtención de un clasificador con SVM basado *Fuzzy C-Means* teniendo como objetivo mejorar el tiempo de entrenamiento de las SVM clásicas
   - El desempeño del algoritmo es muy bueno para conjuntos de datos grandes con dimensiones menores a 20
   - Cuando el tamaño de la dimensión del conjunto de entrada es mayor a 20 el algoritmo presenta algunos problemas para un correcto agrupamiento del conjunto de datos de entrada

2 El algoritmo MEB-SVM^2, fue diseñado para trabajar en conjuntos de datos de gran tamaño, obteniendo los siguientes resultados:

- Un clasificador con SVM basado en Esferas de Cerradura Mínima teniendo como objetivo mejorar el tiempo de seccionamiento del conjunto de datos de entrada
- El desempeño del algoritmo es muy bueno en conjuntos de datos grandes con alta dimensionalidad
- El tiempo de entrenamiento total es reducido significativamente con respecto al algoritmo FCM-SVM^2.

3 El algoritmo RS-SVM^2 fue diseñado para trabajar en conjuntos de datos de gran tamaño obteniendo los siguientes resultados:

- Un clasificador con SVM basado en Esferas de Cerradura Mínima
- El algoritmo propuesto emplea un método de selección aleatoria evitando agrupar el conjunto de datos de entrada.
- El desempeño del algoritmo es muy bueno en conjuntos de datos grandes con alta dimensionalidad

- El tiempo de entrenamiento total es reducido aún con respecto al algoritmo MEB-SVM^2

4 El algoritmo de multiclasificación (mMEB-SVM^2) fue diseñado para trabajar en problemas con múltiples clases, los resultados obtenidos fueron los siguientes:

- Un clasificador con SVM basado en Esferas de Cerradura Mínima
- El algoritmo propuesto, no secciona el conjunto de datos de entrada. El algoritmo emplea un método de selección aleatoria.
- El algoritmo mMEB obtiene los pares de vectores soporte con distinta clase más cercanos y crea una esfera entre cada par de vectores soporte, obteniendo todos los puntos entre estos vectores soporte previos en el conjunto de datos original.
- El desempeño del algoritmo es muy bueno en conjuntos de datos grandes con alta dimensionalidad

En este trabajo se proponen algoritmos para clasificación binaria y algoritmos de clasificación para conjuntos de datos con múltiples clases. Los algoritmos propuestos fueron probados con grandes conjuntos de datos y comparados con otras implementaciones representativas del estado del arte de SVM para grandes conjuntos de datos. Los resultados obtenidos muestran que los algoritmos propuestos tienen ventaja sobre las implementaciones más representativas del estado del arte.

8.2. Trabajo Futuro

Como posibles rutas para trabajo futuro se pueden se pueden señalar los siguientes puntos:

1. La selección de los parámetros de los clasificadores basados en kernels es crucial y no es la excepción en los algoritmos propuestos. Es claro, que la selección de

parámetros tiene un efecto significativo sobre el desempeño de los clasificadores propuestos. Una investigación más exhaustiva debe realizarse para encontrar la forma óptima de selección de parámetros.

2. Una de las mayores debilidades de las SVM es que son esencialmente clasificadores binarios. Los enfoques descritos en el Capítulo 6 habilitan a las SVM para trabajar con problemas de multi-clasificación, la revisión de la función de decisión y empleo de otros kernels podría mejorar la precisión en problemas de multi-clasificación.

3. En los algoritmos propuestos se emplean dos etapas de clasificación, empleando una SVM en cada etapa. Sin embargo, es posible mejorar la precisión de clasificación empleando multiples capas de clasificadores SVM en una primera etapa de clasificación y un clasificador en la etapa final.

4. La detección de exones dentro de secuencias de ADN es un problema muy importante en bioinformática, ya que en algunos organismos los intrones en células eucariotas son bastantes y los sitios de empalmes no son bien caracterizados, se requieren técnicas que definan perfectamente la frontera entre una clase y otra. Se han llevado a cabo diferentes experimentos para clasificación de exones e intrones con las diferentes implementaciones desarrolladas y en algunos casos se han obtenido muy buenos resultados. Sin embargo, no existe una investigación que incluya un análisis de la influencia de las características asociadas a secuencias de ADN. Como trabajo futuro, se pretende realizar un estudio de la importancia de estas características, así como la reducción de atributos irrelevantes para una mejor clasificación.

5. Para la detección de exones en secuencias de ADN se requieren técnicas de codificación que ayuden a mejorar la caracterización de las secuencias de ADN, mejorar las técnicas de codificación mejoraría significativamente la precisión de clasificación.

6. Un problema muy importante en visión es la clasificación de texturas. En algunos casos, se requiere un tiempo de respuesta casi inmediato y en la mayoría de ellos resultados muy precisos para aumentar, disminuir o modificar la velocidad de desplazamiento de sistemas mecánicos. Ya se han utilizado los algoritmos desarrollados en esta tesis para clasificación de texturas obteniendo un buen desempeño en algunos de ellos, sin embargo en el estudio futuro se pretende analizar diferentes kernels en busca de mejorar la precisión de clasificación.

8.3. Publicaciones

Durante el trabajo de tesis se publicaron los siguientes artículos:

Publicaciones en revista

1. J. Cervantes, X.Li and W.Yu, "Support vector machine classification for large data sets via minimum enclosing ball clustering," *Neurocomputing*, Vol. 71, Issues 4-6, pp. 611-619 , 2008.

2. Jair Cervantes, Farid Garcia, Xiaoou Li and Wen Yu, "A data reduction algorithm based on SVM", *Research in Computer Science*, Vol, 38, pp 201-211. 2008.

3. Jair Cervantes, Xiaoou Li and Wen Yu, "Two Stages of SVM Classification for Large Data Sets Via Clustering Techniques", submitted in *Pattern Recognition Letters*

Publicaciones en congresos

1. J. Cervantes, X.Li and W.Yu, "Multi-Class Support Vector Machines for Large Data Sets via Minimum Enclosing Ball Clustering," *ICEEE 2007. 4th International Conference on Electrical and Electronics Engineering,* pp. 146 - 149, 2007

2. Xiaoou Li, Jair Cervantes, and Wen Yu, Two-Stage SVM Classification for Large Data Sets via Randomly Reducing and Recovering Training Data, *IEEE International Conference on Systems, Man, and Cybernetics, SMC'07*, pp. 3633-3638, 2007.

3. Jair Cervantes, Xiaoou Li and Wen Yu, "SVM Classification for Large Data Sets by Considering Models of Classes Distribution", *6th Mexican International Conference on Artifical Intelligent (MICAI 2007)*, Aguascalientes, Mexico, 51-60, 2007

4. Jair Cervantes,Xiaoou Li, and Wen Yu, Multi-Class SVM for Large Data Sets Considering Models of Classes Distribution, *The 2008 International Conference on Data Mining*, Las Vegas, USA, 2008

5. Jair Cervantes, Xiaoou Li and Wen Yu, " Support Vector Classification for Large Data Sets by Reducing Training Data with Change of Classes" In *The 2008 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2008)*, pp. Singapore.

6. Jair Cervantes, Xiaoou Li and Wen Yu and Rosa Bermudez, "Intron/Exon Classification using SVM and Decision Trees", *First International Congress on Biotechnology and Bioengineering (1ICBB 2008)*, Mexico D.F., 2008

7. Jair Cervantes, Xiaoou Li and Wen Yu, "Splice Site Detection in DNA Sequences Using a Fast Classification Algorithm", Aceptado para publicación en *The 2009 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2009).*

Capítulos de libro

1. J.Cervantes, X.Li and W.Yu, Support Vector Machines Classification Based on Fuzzy Clustering for Large Data Sets, *MICAI 2006: Advances in Artificial Intelligence, Springer-Verlag, Lecture Notes in Computer Science (LNCS),* Vol. 4293, pp. 572-582, 2006.

Bibliografía

[1] Ambwani Tarun, "Multi Class Support Vector Machine Implementation To Intrusion Detection," *In:2003 International Joint Conference on Neural Network*, pp. 2300- 2305, 2003

[2] Awad M., Khan L., Bastani F. y Yen I. L., An Effective Support Vector Machines (SVMs) Performance Using Hierarchical Clustering, *In Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04),* pp. 663- 667, 2004

[3] G. Phanendra Babu and M. Narasimha Murty, A near-optimal initial seed value selection in K-means algorithm using a genetic algorithm, *Pattern Recognition Letters*, Vol. 14, No. 10, pp. 763–769, 1993.

[4] Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via coresets. *In Proceedings of the 34th Symposium on Theory of Computing (STOC '02)*, pp. 250–257, 2002

[5] Balcazar J. L., Dai Y. y Watanabe O.,Provably Fast Training Algorithms for support vector machine, *In Proceedings of the 1st IEEE Int. Conf. on Data Mining*, pp. 43-50, 2001.

[6] L. Bottou and C.-J. Lin. Support Vector Machine Solvers. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, Large Scale Kernel Machines, pages 1–28. MIT Press, 2007.

[7] J.Cervantes, X.Li and W.Yu, Support Vector Machine Classification Based on Fuzzy Clustering for Large Data Sets, *MICAI 2006: Advances in Artificial Intelligence, Srpinger-Verlgag, Lecture Notes in Computer Science (LNCS),* vol. 4293, 572-582, 2006

[8] Cervantes, J., Li, X., Yu, W., and Li, K. Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing* Vol. 71, No. 4-6, pp. 611–619, 2008

[9] C-C.Chang and C-J.Lin, LIBSVM: a library for support vector machines, *http://www.csie.ntu.edu.tw/~cjlin/libsvm*, 2001.

[10] Chen Yixin, Wang James Z. Support Vector Learning for Fuzzy Rule-Based Classification Systems, *IEEE Transactions on Fuzzy Systems*, Vol. 11, No. 6, pp. 716-728, December 2003

[11] Chen J.-H. and Chen Ch.-S., Reducing SVM classification time using multiple mirror classifiers, *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, Vol. 34, Issue: 2, pp. 1173- 1183, April 2004

[12] R.Collobert, S.Bengio, and Y.Bengio, A Parallel Mixture of SVMs for Very Large Scale Problems, *Neural Computation*, Vol. 14, No.5, pp. 1105-1114, 2002.

[13] Collobert R. y Bengio S., SVMTorch: support vector machine for large regresion problems. *Journal of Machine Learning Research*, Vol.1, pp. 143-160, 2001.

[14] P-H.Chen, R-E.Fan and C-J.Lin, A Study on SMO-Type Decomposition Methods for Support Vector Machines, *IEEE Trans. Neural Networks*, Vol.17, No.4, pp. 893-908, 2006

[15] N.Cristianini, J.Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods , *Cambridge University Press*, 2000

[16] Alan Wee-Chung Liew and Yonghui Wu and Hong Yan, Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy, IEEE Trans. Pattern Anal. Mach. Intell., Vol. 27, No. 8, pp.1226–1238, 2005

[17] Yukun Bao, Rui Zhang and Sven F. Crone, Fuzzy Support Vector Machines Regression for Business Forecasting: An Application, *In Lecture Notes in Computer Science,*Volume 4223, pp. 1313-1317, 2006.

[18] P. Bhattacharya, M. Rahman and B.C. Desai, Image Representation and Retrieval Using Support Vector Machine and Fuzzy C-means Clustering Based Semantical Spaces, *In 18th International Conference on Pattern Recognition, ICPR 2006*, Vol. 1, pp. 929 - 935, 2006.

[19] Daniael B. y Cao D.,Training support vector machine Using Adaptive Clustering, *in Proceedings of SIAM Int. Conf on Data Mining 2004*, pp. 126-137, 2004

[20] Burges, C. J. 1998. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.* Vol.2, No. 2, pp. 121-167,1998.

[21] Courant R. y Hilbert D., *Methods of mathematical physics,* Vol. 1, Interscience London, England, 1953.

[22] J-X.Dong,A.Krzyzak,and C.Y.Suen, Fast SVM Training Algorithm with Decomposition on Very Large Data Sets, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 27, No.4, pp. 603-618, 2005.

[23] Durbha S.S. y King R.L., Knowledge Mining in Earth Observation Data Archives: A Domain Ontology Perspective, *In Proceedings of IEEE International Geoscience and Remote Sensing Symposium*, Vol. 1, pp.1-13, September 2004

[24] Fan H. y Ramamohanarao K., A Weighting Scheme Based on Emerging Pattenrs for Weighted Support Vector Machines, *In Proceedings IEEE International Conference on Granular Computing (GrC 2005),* pp. 435-440, 2005

[25] Feng Hong-Hai, Liao Ming-Yi, Chen Guo-Shun, Yang Bing-Ru, Chen Yu-Mei, SVM and reduction-based two algorithms for examining and eliminating mistakes in inconsistent examples, *ICMLC'04*, Vol.4, pp.2189-2192, 2004

[26] Fickett, J. W., Finding genes by computer: the state of the art, *Trends in Genetics*, Vol 12, No. 8, pp. 316–320, 1996

[27] G.Folino, C.Pizzuti and G.Spezzano, GP Ensembles for Large-Scale Data Classification, *IEEE Trans. Evol. Comput.*, Vol.10, No.5, pp. 604–616, 2006.

[28] Fu, X., Ong, CJ, Keerthi, S., Hung, GG & Goh,. L. Extracting the Knowledge Embedded in Support Vector Machines, *In International Joint Conference on Neural Networks (IJCNN 2004)*, Vol. 1, pp. 291–296, 2004.

[29] M. Girolami, Mercer kernel based clustering in feature space, *IEEE Trans. Neural Networks*, Vol. 13, No. 3, pp. 780–784, May 2002.

[30] B.V.Gnedenko, Y.K.Belyayev, and A.D.Solovyev, Mathematical Methods of Reliability Theory. *NewYork: Academic Press*, 1969

[31] T.Hastie and R.Tibshirani, Discriminant adaptive nearest neighbor classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No 6, pp. 607-616, 1996

[32] Simon Haykin, Neural networks: a comprehensive foundation, Prentice-Hall, Inc, 1999

[33] Dug Hun Hong and Changha Hwang, Support vector fuzzy regression machines, *In Fuzzy Sets and Systems,* Vol. 138, Issue 2, pp. 271-281, 2003.

[34] Hsien, Jung Chiang; Hao, Pei-Yi, Support vector learning mechanism for fuzzy rule-based modeling: a new approach, *Fuzzy Systems, IEEE Transactions on* ,Vol. 12, Issue 1, pp. 1 - 12, Feb. 2004

[35] G.B. Huang, K.Z. Mao, C.K. Siew and D.-S. Huang, Fast Modular Network Implementation for Support Vector Machines", *IEEE Trans. on Neural Networks*, Vol. 16, pp. 1651-1663, 2005

[36] Joachims T.,Making large-scale support vector machine learning practical. *In A.S.B. Scholkopf, C. Burges, editor, Advances in Kernel Methods: support vector machine* . pp.169-184, 1999.

[37] Jones, D., & Watkins, C. Comparing kernels using synthetic DNA and genomic data. *Technical report. Department of Computer Science*, University of London, UK, 2000.

[38] Jun H. y Houkun H., An aLgorithm for Text Categorization with SVM, *Proceedings of IEEE TENCON*, pp. 47-50, 2002.

[39] Kam Ho T. y Kleinberg E.. Building projectable classifiers of arbitrary complexity. *In Proceedings of the 13th International Conference on Pattern Recognition*, pp. 880-885, Vienna, Austria,1996

[40] Karush, W. Minima of functions of several variables with inequalities as side conditions. *Master's thesis, Department of Mathematics*, University of Chicago, 1939.

[41] Keerthi S. S., Shevade S.K., Bhattachayya C., y Murthy K.R.K., *Improvements to platt's SMO algorithm for SVM classifier design,* Tech. Rep. CD-99-14, National University of Singapore, 1999.

[42] Khan L., Awad M and Thuraisingham B., A new intrusion detection system using support vector machines and hierarchical clustering, *The VLDB Journal*, Vol. 16, No. 4, pp. 507-521, 2007.

[43] S.W.Kim; Oommen, B.J.; Enhancing prototype reduction schemes with recursion: a method applicable for "large"data sets, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, Vol. 34, Issue 3, pp. 1384 - 1397, 2004.

[44] Kuhn, H. W., and Tucker, A. W. Nonlinear programming. *In Proceedings of the Second Berkeley Symposium* , J. Neyman, Ed., University of California Press, Berkeley, pp. 481-492, 1951.

[45] P. Kumar, J. S. B. Mitchell, and A. Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *ACM Journal of Experimental Algorithmics*, Vol. 8, pp.1-29, January 2003.

[46] Lebrun, G.; Charrier, C.; Cardot, H.; SVM training time reduction using vector quantization, *Proceedings of the 17th International Conference on pattern recognition,* Vol. 1, pp. 160 - 163, 2004

[47] Y.Leung,J-H.Ma and W-X.Zhang, A New Method for Mining Regression Classes in Large Data Sets, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.23, No.1, pp. 5-21, 2001

[48] Liew, A. W., Wu, Y., and Yan, H. Selection of Statistical Features Based on Mutual Information for Classification of Human Coding and Non-coding DNA Sequences. *In Proceedings of the Pattern Recognition, 17th international Conference on (Icpr'04)* Vol. 3, pp. 766-769, 2004

[49] A.W.C. Liew. and H. Yan and M. Yang, Data mining for Bioinformatics, *Bioinformatics technologies, Springer*, 2005

[50] K.Li; H.K.Huang; Incremental learning proximal support vector machine classifiers, *Proceedings. 2002 International Conference on machine learning and cybernetics,* Vol. 3, pp. 1635-1637, 2002

[51] C-T.Lin,C-M.Yeh,S-F.Liang,J-F.Chung and N.Kumar, Support-Vector-Based Fuzzy Neural Network for Pattern Classification, *IEEE Transactions on Fuzzy Systems*, Vol.14, No.1, pp. 31-41, 2006.

[52] Tao Li, Shenghuo Zhu, Mitsunori Ogihara, Using Discriminant Analysis for Multi-class Classification, *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03),* Vol. 10, No. 4, pp. 453-472

[53] Luo F., Khan L., Bastani F., Yen I., y Zhou J., A Dynamical Growing Self Organizing Tree (DGSOT) for Hierarquical Clustering Gene Expression Profiles, *Bioinformatics*, Vol. 20, Issue 16, pp. 2605-2617, 2004.

[54] M.E.Mavroforakis and S.Theodoridis, A Geometric Approach to Support Vector Machine(SVM) Classification, *IEEE Transactions on Neural Networks*, Vol.17, No.3, pp. 671-682, 2006.

[55] Mitra S. y Hayashi Y., Neuro-fuzzy rule generation: Survey in soft computing framework, *IEEE Transactions on Neural Networks*, Vol. 11, No.3 pp. 748-768, 2000.

[56] Mitra P., Murthy C.A. y Pal S.K., A Probabilistic Active Support Vector Learning Algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 26, No. 3, pp. 413-418, March 2004.

[57] Naenna, T.; Bress, R.A.; Embrechts, M.J., DNA classifications with self-organizing maps (SOMs), *Soft Computing in Industrial Applications*, pp. 151-154, 2003

[58] Nahla H. Barakat, Data mining task from Support Vector Machines: A Sequential Covering Approach, IEEE Trans. on Knowledge and Data Engineering, Vol. 19, No.6, pp.729–741, 2007

[59] Laskaris, N. A. and Zafeiriou, S. P. Beyond FCM: Graph-theoretic post-processing algorithms for learning and representing the data structure. *Pattern Recognigtion.* Vol. 41, No. 8, pp. 2630-2644, 2008

[60] Núñez Haydemar, Angulo Cecilio, Català Andreu, Rule Extraction from Support Vector Machines, *In ESANN'2002 Proceedings - European Symposium on Artificial Neural Networks*, pp. 107-112, April 2002

[61] A.Nurnberger, W.Pedrycz , R.Kruse, Data mining tasks and methods: Classification: neural network approaches, *Oxford University Press, Inc., New York, NY*, 2002

[62] Ogura H. and Agata H. and Xie M. and Odaka T. and Furutani H., A study of learning splice sites of DNA sequence by neural networks, *Computers in biology and medicine,* Vol. 27, No. 1, pp. 67-75, 1997

[63] Olson, David L., Delen, Dursun, Advanced Data Mining Techniques, *Springer*, 2008

[64] Onoda T., Murata H. y Yamada S., Relevance Feedback Documente Retrieval using Support Vector Machines, *Active Mining, Second International Workshop, AM 2003,* pp.59-73, 2003

[65] N. Pal and J. Bezdek, On cluster validity for the fuzzy c-means model, *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 3, pp. 370–379, 1995

[66] Pavlov, D.; Mao, J.; Dom, B.; Scaling-up support vector machine using boosting algorithm, *Proceedings of 15th International Conference on pattern recognition,* Vol. 2, pp. 219 - 222, 2000

[67] Platt J., "Fast Training of support vector machine using sequential minimal optimization. *In A.S.B. Scholkopf, C. Burges, editor, Advances in Kernel Methods: support vector machine* . MIT Press, Cambridge, MA 1998

[68] John C. Platt and Nello Cristianini and John Shawe-taylor, Large margin dags for multiclass classification, *Advances in Neural Information Processing Systems*, pp. 547–553, 2000

[69] C.Pizzuti and D.Talia, P-Auto Class: Scalable Parallel Clustering for Mining Large Data Sets, *IEEE Trans. Knowledge and Data Eng.*, Vol.15, No.3, pp. 629-641, 2003.

[70] Daniael Prokhorov. IJCNN 2001 neural network competition. *Slide presentation in IJCNN'01, Ford Research Laboratory*, 2001. *http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf*

[71] Quinlan, J. R. 1993 C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc.

[72] Gunnar Ratsch and Soren Sonnenburg. Accurate Splice Site Detection for Caenorhabditis elegans.,in Kernel Methods in Computational Biology B. Schlkopf, K. Tsuda and J.-P. Vert Editors, MIT press. 2003

[73] Rychetsky, Matthias. Algorithms and Architectures for Machine Learning based on Regularized Neural Networks and Support Vector Approaches, *Shaker Verlag,* 2001

[74] Rui Xu; Wunsch, D., II., Survey of clustering algorithms, *IEEE Transactions on Neural Networks*, Vol. 16, Issue 3, pp. 645-678, May 2005

[75] Ruping Stefan, Incremental Learning with Support Vector Machines, *Proceedings of the 2001 IEEE International Conference on Data Mining*,pp 641-642, 2001

[76] R Safavian, D Landgrebe , A survey of decision tree classifier methodology, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No3, pp. 660-674, 1991

[77] Saketha Nath J., Bhattacharyya C., Murty M. N., Clustering based large margin classification : A scalable approach using SOCP formulation, *Proceedings of the Twelfth ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 674-679, 2006

[78] Sarle W.S. Part 2 of Neural network FAQ. Periodic posting to the usenet news group comp.ai.neural-nets

[79] Schohn G. y Cohn D., Less is more : Active Learning with support vector machine, *In Proceedings. 17th Int. Conf. Machine Learning*, pp.839-846, 2000.

[80] Shawe-Taylor, John, Cristianini Nello. Kernel Methods for Pattern Analysis. *Cambridge University Press*, 2004

[81] Shih L., Rennie D.M., Chang Y. y Karger D.R., Text Bundling: Statistics-based Data Reduction, *In Proc of the Twentieth Int. Conf. on Machine Learning (ICML-2003)*, pp. 696-703, 2003

[82] Silva C. y Ribeiro B., Margin-Based Active Learning and Background Knowledge in Text Mining, *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems*, pp. 8-13, 2004

[83] Soman K.P. , Shyam Diwakar M. , y Madhavdas P., Efficient classification and analysis of Ischemic Heart Disease using Proximal Support Vector Machines based Decision Trees, *IEEE International conference Tencon 2003*, Vol. 1, pp.214-217, 2003

[84] Steinwart I. Sparseness of Support Vector Machines - some asymptotically sharp bounds. *In proceedings of the 16^{th} NIPS Conferenece*, pp 169-184, 2004

[85] Suykens, J.A.K., De Brabanter J., Lukas L., Vandewalle J., Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, Vol. 48, pp 85-105, 2002

[86] P.Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, *Pearson Addison Wesley*, May, 2005

[87] Tang H., Mukomel Y. y Fink Eugene, Diagnosis of Ovarian Cancer Based on Mass Spectra of Blood Samples, *IEEE International Conference on Sistems, Man and Cybernetics,* pp. 3444-3450, 2004

[88] Tong S. y Koller D., Support vector machine active learning with applications to text clasifications, *In Proceedings 17th Int. Conf. Machine Learning*, pp. 999-1006, 2000.

[89] V.Tresp, A Bayesian Committee Machine, *Neural Computation*, Vol.12, No.11, pp. 2719-2741, 2000

[90] I.W.-H.Tsang, J.T.-Y.Kwok, J.M.Zurada, Generalized Core Vector Machines, *IEEE Trans. Neural Networks*, Vol.17, No.5, pp. 1126-1140, 2006

[91] Vapnik V., The Nature of Statistical Learning Theory, *Springer, N.Y.*, 1995

[92] Vapnik V., Statistical Learning Theory, *Springer, N.Y.*, 1998

[93] Wang L.X. y Mendel J.M., Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least Squares Learning, *IEEE Transactions on Neural Networks*, Vol. 3, pp. 807-814, Sept. 1992

[94] Wei L., Qi J.J., y Zhang W.X., Knowledge Discovery of Decision Table Based on Support Vector Machine, *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, Vol. 2, pp. 1195.1200, 2003

[95] A.Uzilov, J.Keegan and D.Mathews, *BMC Bioinformatics, www.pubmedcentral.nih.gov*.

[96] Vishwanathan, S.V.M.; Narasimha Murty, M., SSVM: a simple SVM algorithm, *In Proceedings of the 2002 International Joint Conference on Neural Networks*, Vol.3, pp. 2393-2398, 2002

[97] Van Gestel, T., Suykens, J.A.K., De Moor, B., Vandewalle, J., Bayesian inference for LS-SVMs on large data sets using the Nystrom method, *Proceedings of the 2002 International Joint Conference on neural networks*, Vol. 3, pp. 2779-2784, 2002

[98] Ya Z., Chao-Hsien Ch., Yixin Ch., Hongyuan Z, Xiang J., Splice site prediction using support vector machines with a Bayes kernel, *Expert Systems with Applications*, Vol. 30, No. 1, pp. 73-81, 2006

[99] Yu H., Yang J. y Han Jiawei, Classifying Large Data Sets Using SVMs with Hierarchical Clusters, *in Proceedings of the 9th ACM SIGKDD 2003*, pp. 3006-315, 2003.

[100] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pp. 103–114, June 1996.