UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

UTILIZACIÓN DE SUPPORT VECTOR MACHINES NO LINEAL Y SELECCIÓN DE ATRIBUTOS PARA CREDIT SCORING

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN GESTIÓN DE OPERACIONES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

SEBASTIÁN MALDONADO ALARCÓN

PROFESOR GUÍA:
RICHARD WEBER HAAS

MIEMBROS DE LA COMISIÓN:
VIVIANA FERNÁNDEZ MATURANA
JAIME MIRANDA PINO
PABLO ANDRÉS REY

SANTIAGO DE CHILE
JULIO, 2007

Índice general

Capítulo 1

Introducción

Con la revolución digital capturar información es fácil y almacenarla es extremadamente barato. Pero, ¿para qué almacenar datos? Además de la facilidad y conveniencia, las empresas almacenan datos porque piensan que son un activo valioso en sí mismos. Para los científicos los datos representan observaciones cuidadosamente recogidas de algún fenómeno en estudio. En los negocios, los datos guardan informaciones sobre mercados, competidores y clientes. En procesos industriales recogen valores sobre el cumplimiento de objetivos.

Los datos en bruto raramente son provechosos. Su verdadero valor radica en la posibilidad de extraer información útil para la toma de decisiones o la exploración y comprensión de los fenómenos que dieron lugar a los datos. Tradicionalmente, el análisis de estos datos ha sido efectuado mediante técnicas estadísticas. Sin embargo, el incremento en la cantidad de datos y en el número de parámetros hace necesaria la aparición de nuevas metodologías y herramientas para un tratamiento automático de los registros depositados en bases de datos.

El proceso completo de extraer conocimiento a partir de bases de datos se conoce como *KDD* (Knowledge Discovery in Databases) [12]. Este proceso comprende diversas etapas, que van desde la obtención de los datos hasta la aplicación del conocimiento adquirido en la toma de decisiones. Entre esas etapas, se encuentra la que puede considerarse como el núcleo del proceso *KDD* y que se denomina Minería de Datos o Data Mining [12]. Esta fase es crucial para la obtención de resultados apropiados, pues durante la misma se aplica el algoritmo de aprendizaje automático encargado de extraer el conocimiento inherente a los datos. No obstante, esta fase se ve influida en gran

medida por la calidad de los datos que llegan para su análisis desde la fase previa.

Algunos problemas comunes a la mayoría de los algoritmos de aprendizaje automático cuando trabajan con muchos atributos se refieren a tiempos de ejecución muy elevados, pobre capacidad de generalización frente a nuevas observaciones o incomprensión de los resultados obtenidos. Por ello, desde la década de los sesenta, las investigaciones relacionadas con la selección de atributos intentan reducir el espacio de hipótesis de las bases de datos en tareas concretas, en un intento de encontrar subconjuntos de atributos que proporcionen un mejor rendimiento de los algoritmos de aprendizaje.

Por otro lado, el método cuantitativo conocido como Credit Scoring [26] ha sido desarrollado para el problema de asignación de créditos. Credit Scoring es esencialmente una aplicación de las técnicas de clasificación, la cual clasifica a los clientes en diferentes grupos de riesgo. Esto permite a la entidad crediticia cuantificar y manejar el riesgo financiero relacionado con el otorgamiento de créditos, para así tomar decisiones de forma rápida y objetiva.

Recientemente, el modelo de clasificación Support Vector Machines (SVM) [41] ha recibido una creciente atención, debido a interesantes características tales como buena generalización de la aplicación a nuevos objetos, la ausencia de mínimos locales y representación que depende de pocos parámetros. Esta técnica ha probado su buen desempeño en muchas aplicaciones, tales como Credit Scoring [1], predicción de series de tiempo financieras [14], categorización de “Spam” [11] y clasificación de tumores cerebrales [24]. Sin embargo, dentro de los sistemas de SVM hay muy pocos enfoques establecidos para identificar los atributos más importantes para construir una regla discriminante, por lo tanto en esta Tesis se desarrollarán métodos que permitan realizar la tarea de selección de atributos durante el proceso de construcción de la regla discriminante, y se aplicarán a un caso real de Credit Scoring para evaluar sus desempeños.

1.1. Título

Utilización de Support Vector Machines no Lineal y selección de atributos para Credit Scoring.

1.2. Objetivos

1.2.1. Objetivo General

Este trabajo de investigación tiene como objetivo principal el estudio y propuesta de métodos de selección de atributos que se pueda aplicar a bases de datos de elevada dimensión en el marco del aprendizaje supervisado, en concreto para la tarea de clasificación mediante Support Vector Machines (SVM) en su versión no lineal, y aplicar estos métodos para el problema de asignación de créditos (Credit Scoring) de una entidad financiera nacional.

1.2.2. Objetivos Específicos

Los objetivos específicos son:

(i) Analizar el trabajo propuesto por R. Montoya [28] para la selección de atributos y SVM en su versión lineal, y estudiar su posible aplicación al caso no lineal.

(ii) Comprender y describir los fundamentos del método SVM y sus principales áreas de utilización.

(iii) Mostrar los beneficios de las metodologías propuestas.

(iv) Extraer reglas que permitan aportar interpretabilidad a la solución de un problema de clasificación de Credit Scoring.

1.3. Metodología de la Tesis

La metodología que se utilizará en el desarrollo de la Tesis contemplará los siguientes aspectos:

- Estudio de artículos científicos y libros relacionados con los temas de selección de atributos, análisis estadístico, Support Vector Machines, Data Mining en general y técnicas usadas para resolver el problema de Credit Scoring.

- Utilización de técnicas de programación matemática y optimización para el diseño de nuevas metodologías.

- Implementación de las metodologías propuestas y testeo en un problema real de Credit Scoring.

- Revisión y comparación de los resultados con distintas técnicas conocidas de la literatura.

- Extracción de reglas a partir del resultado de la clasificación, que le permitan a la entidad financiera tomar decisiones adecuadas.

Capítulo 2

Aprendizaje automático supervisado

Los grandes avances en las tecnologías de la información han provocado un aumento sin precedentes tanto en el volumen de datos como en su flujo. Códigos de barra, tarjetas electrónicas, sensores remotos, transacciones bancarias, satélites espaciales o el descifrado del código genético humano son ejemplos de la necesidad de filtrar, analizar e interpretar volúmenes de datos que se miden en terabytes.

A continuación se introduce el concepto “aprendizaje automático supervisado” [12] en relación con los de minería de datos y extracción de conocimientos. En las siguientes secciones se tratan con detalles aspectos como la representación de los datos, la clasificación, la evaluación del rendimiento de un clasificador, y la descripción de la preparación de los datos para la minería.

Descubrimiento de Conocimiento en Bases de Datos

Desde hace más de dos décadas se vienen desarrollando y utilizando complejos algoritmos para la extracción de patrones útiles en grandes conjuntos de datos. Gran parte de esta información representa transacciones o situaciones que se han producido, siendo útil no sólo para explicar el pasado, sino para entender el presente y predecir la información futura. En muchas ocasiones, el método tradicional de convertir los datos en conocimiento consiste en un análisis e interpretación realizada de forma manual por especialistas en la materia estudiada. Esta forma de actuar es lenta, cara y altamente subjetiva. De hecho, la enorme cantidad de datos desborda la capacidad humana de comprenderlos y el análisis manual hace que las decisiones se tomen según la intuición de los especialistas.

A finales de la década de los 80, la creciente necesidad de automatizar todo este proceso inductivo abre una línea de investigación para el análisis inteligente de datos. Al conjunto de métodos matemáticos y técnicas de software para análisis inteligente de datos y búsqueda de regularidades y tendencias en los mismos, aplicados de forma iterativa e interactiva, se denominaron técnicas de Data Mining [12]. Su nombre proviene de las similitudes encontradas entre buscar valiosa información de negocio en grandes bases de datos y minar una montaña para encontrar una veta de metales valiosos. Su tarea fundamental es la de encontrar modelos inteligibles a partir de los datos, y para que el proceso sea efectivo debe ser automático, generando patrones que ayuden en la toma de decisiones beneficiosas para la organización.

Para obtener conclusiones válidas y útiles al aplicar Minería de Datos, es necesario complementar este proceso con una adecuada preparación de los datos previa al proceso de minería y un análisis posterior de los resultados obtenidos. Así, es posible afirmar que el proceso de minería de datos pertenece a un esquema más amplio, reflejado en la Figura 2.1, denominado extracción o descubrimiento de conocimiento en bases de datos (KDD, Knowledge Discovery in Databases [12]). Se puede intuir que el KDD es un campo multidisciplinar, donde las principales áreas contribuyentes son el aprendizaje automático, las bases de datos y la estadística.

El aprendizaje automático es el área de la inteligencia artificial que se ocupa de desarrollar técnicas capaces de aprender, es decir, extraer de forma automática conocimiento subyacente en la información [12]. Constituye junto con la estadística el corazón del análisis inteligente de los datos. Los principios seguidos en el aprendizaje automático y en la minería de datos son los mismos: la máquina genera un modelo a partir de ejemplos y lo usa para resolver el problema.

Uno de los modelos de aprendizaje más estudiado es el aprendizaje inductivo, que engloba todas aquellas técnicas que aplican inferencias inductivas sobre un conjunto de datos para adquirir conocimiento inherente a ellos [12].

En general, las tareas de un proceso de aprendizaje, al igual que las de uno de minería, pueden ser clasificadas en dos categorías: descriptivas y predictivas.

Las primeras describen el conjunto de datos de una manera resumida y concisa, presentando propiedades generales e interesantes de los datos. Por otro lado, las tareas predictivas construyen uno o varios modelos que realizan inferencia sobre el conjunto de entrenamiento para intentar predecir el comportamiento de nuevos datos. La variedad de términos existente en la literatura especializada para nombrar las tareas y las técnicas utilizadas para desempeñar cada una de ellas, hace que, a veces, sea difícil aclararse con los nombres. Se suele utilizar el término aprendizaje supervisado para los métodos predictivos y aprendizaje no supervisado para el resto. En este trabajo, el aprendizaje siempre será entendido como supervisado, donde los casos pertenecientes al conjunto de datos tienen asignada una clase o categoría a priori, siendo el objetivo encontrar patrones o tendencias de los casos pertenecientes a una misma clase.

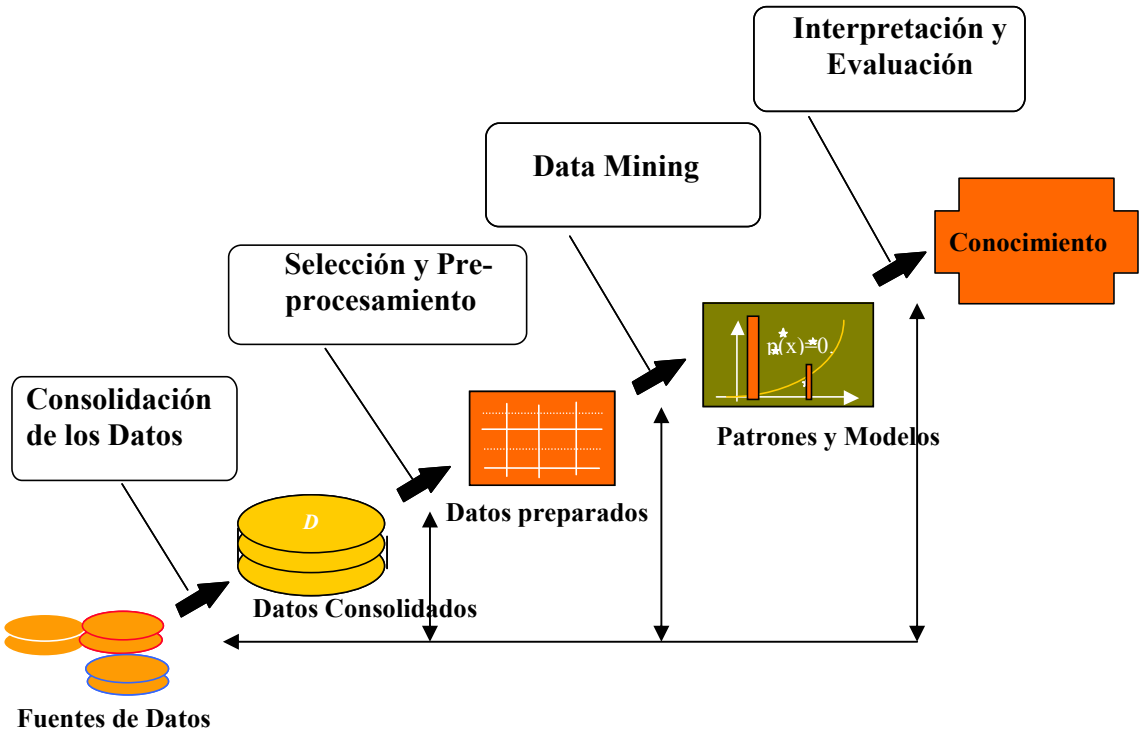

Figura 2.1: Proceso KDD

2.1. Preparación de los datos

La utilidad de la extracción de información de los datos depende en gran medida de la calidad de éstos. Como se pudo comprobar en la Figura 2.1, existe una fase de preparación de los datos previa a su análisis, donde se realiza una serie de tareas que se describen a continuación.

Pyle [31] indica que el propósito fundamental de esta fase es el de manipular y transformar los datos en bruto, de manera que la información contenida en el conjunto de datos pueda ser descubierta, o más fácilmente accesible.

Dado que en muchas ocasiones los datos provienen de diferentes fuentes, pueden contener valores impuros (incompletos, con ruido e inconsistentes), pudiendo conducir a la extracción de patrones poco útiles. Además, se puede reducir el conjunto de datos (selección de características y de instancias), mejorando la eficiencia del proceso de minería de datos posterior. También existe la posibilidad de recuperar información incompleta, eliminar *outliers*, resolver conflictos, etc., generando un conjunto de datos de calidad, que conduciría a mejores patrones. La preparación o preprocesamiento de datos engloba a todas aquellas técnicas de análisis de datos que permiten mejorar la calidad de un conjunto de datos, de modo que los métodos de extracción de conocimiento puedan obtener mayor y mejor información (mejor porcentaje de clasificación, reglas con más completitud, etc.) [47]. La lista de tareas que se incluyen en esta fase, se pueden resumir en cuatro: recopilación de datos, limpieza, transformación y reducción.

2.1.1. Recopilación

Para poder comenzar a analizar y extraer información útil de los datos es preciso, en primer lugar, disponer de ellos. Esto en algunos casos puede parecer trivial, partiendo de un simple archivo de datos, sin embargo en otros, es una tarea muy compleja donde se debe resolver problemas de representación, de codificación e integración de diferentes fuentes para crear información homogénea.

2.1.2. Limpieza

En esta fase se resuelven conflictos entre datos, comprobando problemas de ruido, valores ausentes y *outliers* (inconsistencias o valores fuera de rango) [19]. La presencia de estos problemas en los datos es relativamente frecuente, debido principalmente a fallos cometidos durante el proceso de adquisición de los datos, sea manual o automático. Aunque algunos métodos solventan estos problemas durante el proceso de aprendizaje, es común aplicar alguna técnica que trate estos ejemplos antes de ofrecerlos al algoritmo de Minería de Datos. La técnica de tratamiento de errores más simple, aunque también la menos recomendable, consiste en eliminar aquellos ejemplos que presenten algún atributo sin valor o inconsistente. El mayor inconveniente de esta técnica es que se podría eliminar información útil para el aprendizaje contenida en los atributos correctos. Para poder mantener los ejemplos en el conjunto de datos, habría que rellenar estos valores con algún valor válido. Una solución sencilla es asignar una constante, por ejemplo “desconocido”, si el atributo es discreto, o 1, si es continuo [19]. Aunque esta solución es también muy simple y no elimina información, el algoritmo de aprendizaje podría interpretar erróneamente esas constantes y entender que son valores interesantes. Por esta razón, es recomendable sustituir las ausencias por valores cuya influencia en el aprendizaje sea mínima. En este sentido, la media o la moda, dependiendo si el atributo es continuo o discreto respectivamente, pueden ser valores más apropiados que una constante. Para que el valor de sustitución no sea único para todos los ejemplos con ausencias en un mismo atributo, la media o la moda no se calcula a partir de todos los datos, sino considerando sólo aquellos ejemplos que tienen la misma clase que el que se pretende completar. Finalmente, una técnica más precisa, aunque también más costosa computacionalmente, consiste en sustituir los errores por el valor más probable, aplicando algún clasificador (regresión, clasificador Bayesiano o inducción de árboles de decisión) para predecir dicho valor [31].

2.1.3. Transformación

En ocasiones, la forma en que viene dada la información originalmente no es la más adecuada para adquirir conocimiento a partir de ella. En esas situaciones se hace necesaria la aplicación algún tipo de trasformación para adecuar los datos

al posterior proceso de aprendizaje, como por ejemplo una normalización o cambio de escala, discretización, generalización o extracción de atributos. Esta última trasformación está estrechamente relacionada con la selección de características detallada más adelante y consiste en construir nuevos atributos a partir de combinaciones de los originales.

Se considera una técnica de transformación aquella destinada a modificar los datos para mejorar el proceso de aprendizaje, y no a corregir errores en los mismos. Como ejemplo de necesidad de transformación en los datos, se puede observar la situación que se plantea a continuación. Un gran número de algoritmos de aprendizaje operan exclusivamente con espacios discretos, sin embargo, muchas bases de datos contienen atributos de dominio continuo, lo que hace imprescindible la aplicación previa de algún método que reduzca la cardinalidad del conjunto de valores que estas características pueden tomar, dividiendo su rango en un conjunto finito de intervalos. Esta trasformación de atributos continuos en discretos se denomina discretización. Menos frecuente es la transformación inversa denominada numerización.

2.1.4. Reducción

Los investigadores dedicados al aprendizaje automático supervisado, y concretamente, al estudio de algoritmos que produzcan conocimiento en alguna de las representaciones usuales (listas de decisión, árboles de decisión, reglas de asociación, etc.) suelen realizar las pruebas con bases de datos estándares y accesibles a toda la comunidad científica (la gran mayoría de ellas de reducido tamaño), con objeto de verificar los resultados y validarlos con independencia.

No obstante, y una vez asentadas estas propuestas, algunos de estos algoritmos sufren modificaciones orientadas a problemas específicos, los cuales contienen una cantidad de información muy superior (decenas de atributos y decenas de miles de ejemplos) a la de las bases de datos de prueba. La aplicación de tales técnicas de Minería de Datos es por tanto una tarea que consume una enorme cantidad de tiempo y memoria, aun con la potencia de los ordenadores actuales, que hace intratable la adaptación del algoritmo para solucionar el problema particular.

Es conveniente, entonces, aplicar técnicas de reducción a la base de datos, estando orientadas fundamentalmente hacia dos objetivos: técnicas de editado (reducción del número de ejemplos) y técnicas selección de atributos (eliminar aquellos atributos que no sean relevantes para la información inherente a la base de datos).

2.2. Clasificación

Aprender cómo clasificar objetos a una de las categorías o clases previamente establecidas, es una característica de la inteligencia de máximo interés para investigadores tanto de psicología como de informática, dado que la habilidad de realizar una clasificación y de aprender a clasificar, otorga el poder de tomar decisiones.

El objetivo de la clasificación es generar una función $y = f(\vec{x})$, denominada clasificador, que represente la correspondencia que existe en los ejemplos entre los vectores de entrada y el valor de salida correspondiente, donde $\vec{x} = (x^1, ..., x^n)$ son los valores de la observación en las variables del estudio.

Además, y es nominal, es decir, puede tomar un conjunto de valores $y_1, y_2, ..., y_n$ denominados clases o etiquetas. La función aprendida será capaz de determinar la clase para cada nuevo ejemplo sin etiquetar. Sin lugar a dudas, el éxito de un algoritmo de aprendizaje para clasificación depende en gran medida de la calidad de los datos que se le proporciona.

La aplicación de un algoritmo de aprendizaje tiene como objetivo extraer conocimiento de un conjunto de datos y modelar dicho conocimiento para su posterior aplicación en la toma de decisiones. Existen distintas formas de representar el modelo generado, representación proposicional, árboles de decisión, reglas de decisión, listas de decisión, reglas con excepciones, reglas jerárquicas de decisión, reglas difusas y probabilidades, redes neuronales, están entre las estructuras más utilizadas.

2.3. Técnicas de evaluación

Partiendo de la necesidad de evaluar el desempeño de la clasificación, se dispone de diversas estrategias de validación de un sistema de aprendizaje dependiendo de como se haga la partición del conjunto.

El método de evaluación más básico, la validación simple, utiliza un conjunto de muestras para construir el modelo del clasificador, y otro diferente para estimar el error, con el fin de eliminar el efecto del sobreajuste en los datos (*overfitting*). Entre la variedad de particiones posibles, uno de los más frecuentes es tomar dos tercios de las muestras para el proceso de aprendizaje y el tercio restante para comprobar el error del clasificador (método conocido como *holdout* [46]). El hecho de que sólo se utiliza una parte de las muestras disponibles para llevar a cabo el aprendizaje, es el inconveniente principal de esta técnica, al considerar que se pierde información útil en el proceso de inducción del clasificador. Esta situación se agrava si el número de muestras para construir el modelo es muy reducido, ya sea porque el porcentaje elegido, o por no disponer de más datos.

Otra técnica de evaluación denominada validación cruzada, se plantea para evitar la ocultación de parte de las muestras al algoritmo de inducción y la consiguiente pérdida de información. Con esta técnica el conjunto de datos se particiona en *k* partes mutuamente exclusivas, conteniendo cada una un número similar de ejemplos, indicándose en muchos casos como validación cruzada con *k* partes (*k-fold crossvalidation* [46]). En cada evaluación, se deja uno de los subconjuntos para la prueba, y se entrena el sistema con los *k-1* restantes. Así, la precisión estimada es la media de las *k* tasas obtenidas. En este caso, los subconjuntos de prueba son independientes, no así los de entrenamiento. Por ejemplo, en una validación cruzada con 10 particiones, cada par de subconjuntos de entrenamiento comparten el ochenta por ciento de los datos. La ventaja de la validación cruzada (a diferencia del método *holdout*) es que todos los casos son utilizados en el proceso de aprendizaje y en el de prueba, dando lugar a un estimador con sesgo muy pequeño.

Un caso particular de este método de evaluación es la validación cruzada dejando uno fuera (leaving-one-out crossvalidation), donde *k* es igual al número

de ejemplos del conjunto de datos. En este caso, el clasificador entrena con todas las muestras menos una que deja fuera para realizar la prueba [20].

Además de la elevada varianza de la tasa de aciertos obtenida, el mayor inconveniente de la validación cruzada es el alto costo computacional que supone el aprendizaje del clasificador *k* veces, por lo que no se suele utilizar cuando el número de muestras es elevado o el proceso de inducción del clasificador es computacionalmente costoso.

Capítulo 3

Credit Scoring

En las últimas dos décadas se ha observado un rápido crecimiento tanto en la disponibilidad como en el uso de los créditos de consumo. Hasta hace poco, la decisión de entregar créditos se basaba en el juicio humano para determinar el riesgo de no pago del postulante a crédito en base a los atributos de éste. Sin embargo, el crecimiento de la demanda por crédito ha llevado a desarrollar métodos formales y objetivos para ayudar a los proveedores del crédito a decidir a quien otorgar crédito y a quien no. Este enfoque fue introducido en los años 40 y con los años se ha ido desarrollando significativamente. En los años recientes, la alta competencia de la industria financiera, los avances en la computación y el crecimiento exponencial del tamaño de las bases de datos han llevado a estos métodos a transformarse en una importante herramienta en la industria.

Credit Scoring se define formalmente como un método cuantitativo que se utiliza para predecir la probabilidad de que un aspirante a crédito o un cliente de la entidad crediticia existente deje de pagar el crédito o bien no lo haga una vez que lo reciba [26]. Su objetivo es ayudar a los proveedores de créditos a cuantificar y manejar el riesgo financiero relacionado con el otorgamiento de créditos, para así tomar decisiones de forma rápida y objetiva.

3.1. Beneficios de Credit Scoring

Credit Scoring tiene múltiples beneficios que incumben no sólo a las entidades crediticias, sino también a los beneficiarios del crédito. Por ejemplo, Credit Scoring ayuda a reducir la discriminación porque provee un análisis objetivo del mérito del postulante para recibir un crédito. Esto les permite a los proveedores enfocarse sólo en la información relacionada con la asignación del crédito y así evitar subjetividad. Cuando se le niega un crédito a un cliente en los Estados

Unidos, la “Equal Credit Opportunity Act” exige a la institución financiera proveer las razones de porque fue rechazado. Razones vagas o indefinidas son ilegales, por lo que variables que puedan llevar a discriminación tales como raza, sexo o religión no pueden ser incluidas en estos modelos [26].

Credit Scoring ayuda también a acelerar y a hacer más consistente el proceso de asignación de créditos, permitiendo la automatización del proceso. Esto reduce significativamente la necesidad de intervención humana y por ende los costos asociados a este proceso. Más aún, Credit Scoring puede ayudar a las instituciones financieras a determinar la tasa de interés que deben cobrar a sus clientes y para valorizar portafolios [46]. Clientes con mayor riesgo reciben una tasa de interés más alta y viceversa. Esto ayuda a la entidad a manejar sus cuentas de manera más efectiva y provechosa en términos de utilidades.

Finalmente y gracias a los avances de la tecnología, se han desarrollado modelos para Credit Scoring más efectivos. En consecuencia, entidades crediticias utilizan esta información generada para formular mejores estrategias de cobranza y utilizar sus recursos más eficientemente. En particular, Credit Scoring ayuda a empresas aseguradoras a realizar una mejor predicción de las reclamaciones, controlar el riesgo de manera efectiva y determinar el precio de los seguros de manera adecuada. Esto les permite ofrecer mayor cobertura a más clientes a un precio equitativo, reaccionar rápido ante los cambios del mercado y obtener ventajas competitivas.

3.2. Evaluación de modelos de clasificación de Credit Scoring

3.2.1. Matriz de confusión y los dos tipos de errores

Un modelo de clasificación predice una clase para cada elemento del conjunto de test. Además se conoce la clase real de cada uno. La matriz de confusión ofrece una manera conveniente para comparar la frecuencia de las clases reales y predichas por el modelo para el conjunto de test.

En Credit Scoring, los clientes se clasifican en 2 clases: “buenos” y “malos” (“aceptados” o “rechazados”, “no-default” o “default”). El formato de la matriz de confusión se muestra en la siguiente tabla:

	clase real			
	Número		tasa	
clase predicha	**malo**	**bueno**	**malo**	**bueno**
malo	A	C		$C/(A+C)=\alpha$
bueno	B	D	$B/(B+D)=\beta$	

Tabla 3.1: Matriz de Confusión para el problema de clasificación binario

donde:
A corresponde al número de “malos” clientes predichos correctamente
B corresponde al número de “buenos” clientes predichos incorrectamente
C corresponde al número de “malos” clientes predichos incorrectamente
D corresponde al número de “buenos” clientes predichos correctamente

Un indicador del error que muestra el desempeño global del modelo de clasificación es el siguiente:

$$\text{Tasa error total} = (B+C)/(A+B+C+D) \tag{3.1}$$

Para el problema de Credit Scoring, se deben considerar dos tipos de error [46]:

$$\text{Tasa error Tipo I} = C / (A + C)$$
$$\text{Tasa error Tipo II} = B / (B + D) \tag{3.2}$$

La tasa de error Tipo I se conoce también como tasa de riesgo crediticio, que es la razón de los “malos clientes” categorizados como “buenos”. Cuando esto sucede, el (erróneamente clasificado) cliente “malo” caerá en “default”. De acuerdo a esto, si la institución tiene una tasa alta, significa que la política de otorgamiento de crédito es muy generosa, la institución se expone a riesgo crediticio.

La tasa de error Tipo II es también llamada indicador ß o riesgo comercial, y es la razón de los "buenos" clientes clasificados como "malos". Si esto sucede, el (erróneamente clasificado) cliente "bueno" será rechazado, por ende la entidad tiene un costo de oportunidad causado por la pérdida de "buenos" clientes. Si la institución tiene una tasa ß alta, su política de otorgamiento de crédito es restrictiva y posiblemente esté perdiendo participación de mercado. Por ende se expone a un riesgo comercial.

Por simplicidad, la Tasa de Error Total se usa frecuentemente como criterio para comparar diferentes modelos de clasificación. El supuesto detrás de esto es que ambos tipos de errores tienen la misma importancia. La relevancia de los tipos de errores es obviamente diferente en el problema de Credit Scoring, porque la predicción correcta de "malos" clientes es más importante ya que éstos tienen un mayor costo asociado. La Tasa de Error Total no es un criterio apropiado para medir el desempeño de un modelo de clasificación.

3.2.2. Función de Costos

Un modelo de clasificación óptimo es aquel que minimiza las pérdidas futuras esperadas cuando es usado para clasificar clientes para crédito. Bajo el supuesto de clasificación binaria, la pérdida dependerá de los costos de ambos tipos de errores de clasificación. La función de costos es la siguiente:

$$\text{Costo Esperado} = P_m\, C_{tipoI}\, \alpha + P_b\, C_{tipoII}\, ß \qquad (3.3)$$

donde P_b y P_m son las proporciones de la población de "buenos" y "malos" clientes y C_{tipoI} y C_{tipoII} los costos unitarios de errores Tipo I y Tipo II respectivamente.

La elección de C_{tipoI} y C_{tipoII} tiene un efecto importante en la evaluación del modelo de clasificación. Sin embargo, la estimación de estos costos suele ser un problema complejo. La razón es que los factores que afectan estos costos son difíciles de cuantificar. Por ejemplo, en Credit Scoring C_{tipoI} es el costo de darle crédito a un "mal" cliente. C_{tipoII} es el costo de oportunidad de rechazar a un "buen" cliente.

En la práctica, las decisiones de crédito siguen un proceso complejo, en el cual se involucran muchos factores que afectan los costos por clasificación incorrecta. Debido a estas dificultades e incertidumbres, las instituciones crediticias usualmente no son capaces de estimar estos costos de manera precisa. Algunos casos que suceden en la realidad son los siguientes:

- Cuando un crédito se otorga por más de un año, los costos deben ser calculados de manera dinámica en multi-períodos, de manera que reflejen el flujo de dinero en diferentes años.

- Cuando un cliente es rechazado debido a los resultados del modelo, es re-examinado por analistas de crédito y se le otorga el crédito, posiblemente bajo nuevas condiciones. El costo de re-examinación y la probabilidad de que la decisión sea la correcta debería ser considerada en la función de costos.

- Cuando un cliente falla en cancelar su deuda a tiempo, pero finalmente la paga gracias a medidas de recolección. El costo de esta recolección debería ser considerado.

Capítulo 4

Selección de Atributos

En los últimos años se ha producido un importante crecimiento de las bases de datos en todas las áreas del conocimiento humano. Este incremento es debido principalmente al progreso en las tecnologías para la adquisición y almacenamiento de los datos. Teóricamente, el tener más atributos daría más poder discriminatorio. Sin embargo, la experiencia con algoritmos de aprendizaje ha demostrado que no es siempre así, detectándose algunos problemas: tiempos de ejecución muy elevados, aparición de muchos atributos redundantes o irrelevantes, y la degradación en el error de clasificación.

En la selección de características se intenta escoger el subconjunto mínimo de atributos de acuerdo con dos criterios: que la tasa de aciertos no descienda significativamente y que la distribución de la clase resultante sea lo más semejante posible a la distribución de clase original considerando todos los atributos. En general, la aplicación de la selección de características ayuda en todas las fases del proceso de Minería de Datos para el descubrimiento de conocimiento.

La intención de este capítulo es la de situar el preprocesado de los datos, y más concretamente la selección de características, dentro del proceso de extracción de conocimiento.

4.1. Definición

Partiendo de la premisa de que en el proceso de selección de atributos se escoge un subconjunto de atributos del conjunto original, este proceso pretende

elegir atributos que sean relevantes para una aplicación y lograr el máximo rendimiento con el mínimo esfuerzo. Se debe tener en cuenta que los atributos irrelevantes y redundantes existentes en una base de datos pueden tener un efecto negativo en los algoritmos de clasificación [3,23]: (1) Al tener más atributos, normalmente implica la necesidad de tener más instancias para garantizar la fiabilidad de los patrones obtenidos (variabilidad estadística entre patrones de diferente clase). Por consiguiente, el algoritmo de clasificación entrenado con todos los datos tardará más tiempo. (2) Los atributos irrelevantes y los redundantes pueden "confundir" al algoritmo de aprendizaje, en el sentido de incorporar patrones que no guardan relación con el fenómeno a describir, por lo que en general, el clasificador obtenido es menos exacto que otro que aprenda sólo de datos relevantes. (3) Además, con la presencia de atributos redundantes o de irrelevantes, el clasificador obtenido será más complejo, dificultando el entendimiento de los resultados. (4) La presencia de estos atributos provoca un mayor costo computacional para generar la función de clasificación ya que el tener más datos implica un mayor tiempo de procesamiento de ellos. (5) Finalmente, la reducción de características se podría tener en cuenta en futuras capturas de datos, reduciendo el costo de almacenamiento y posiblemente el económico.

En resumen, el resultado obtenido al aplicar técnicas de selección de atributos sería:

Menos datos → los clasificadores pueden aprender más rápido.
Mayor exactitud → el método entrega mejores resultados.
Resultados más simples → más fácil de entender.
Menos atributos → menor costo de recolección y almacenamiento de datos.

Se puede concluir que la selección es efectiva en eliminar atributos irrelevantes y redundantes, incrementando la eficiencia en las tareas de minería, mejorando el rendimiento y la comprensión de los resultados [3,22].

4.2. Proceso general

La selección de atributos se puede considerar como un problema de búsqueda [21] en un espacio de estados, donde cada estado corresponde con un subconjunto de atributos, y el espacio engloba todos los posibles subconjuntos que se pueden generar. En un conjunto de datos donde el número de atributos sea cuatro ($n = 4$), el espacio total lo componen dieciséis subconjuntos (2^n). El proceso de selección de atributos puede entenderse como el recorrido de dicho espacio hasta encontrar un estado (combinación de atributos) que optimice alguna función definida sobre un conjunto de atributos.

En general, un algoritmo de selección de atributos se basa en dos pasos básicos: generación de subconjuntos y evaluación de subconjuntos. En la generación de nuevos subconjuntos se define un punto de partida, para que siguiendo una estrategia se recorra el espacio de búsqueda hasta que se cumpla un criterio de parada. Aunque para finalizar el proceso de búsqueda se utilice la función de evaluación, se comentarán las distintas opciones de parada existentes en el proceso de generación de subconjuntos [23].

4.2.1. Estudios preliminares

Existen bastantes referencias de trabajos relacionados con la selección de atributos, pero además, se han realizado estudios sobre diversos aspectos de la selección de atributos (técnicas de búsqueda, medidas de bondad de los atributos, etc.), donde se agrupan los distintos algoritmos existentes en la bibliografía general.

Langley [21] divide las funciones de evaluación en dos categorías: filtro y envolvente (Wrapper). En la primera categoría se incluyen los algoritmos en los que la selección de atributos se realiza como un preproceso a la fase de inducción y por tanto de manera independiente, por lo que puede entenderse como un filtrado de los atributos irrelevantes y redundantes. Por otro lado, en los métodos de tipo envolvente [17], la selección de atributos y los algoritmos de aprendizaje no son elementos independientes, ya que la selección de los

atributos hace uso del proceso de inducción para evaluar la calidad de cada conjunto de atributos seleccionados en cada momento.

Posteriormente, Blum y Langley [3] establecen una clasificación de las funciones de evaluación utilizando también como criterio la dependencia que existe entre el proceso de selección y el de inducción, agrupándolas en tres categorías. Además de las anteriores, están los algoritmos empotrados (Embedded), donde el inductor cuenta con su propio algoritmo de selección de atributos, como ocurre en los algoritmos que generan árboles de decisión, que utilizan sólo aquellos atributos necesarios para obtener una descripción consistente con el conjunto de aprendizaje. En la última categoría se encuadran los esquemas de selección basados en la ponderación de los distintos atributos. Un ejemplo de este esquema de selección puede observarse en las redes neuronales. Doak [10] establece tres categorías para las estrategias de búsqueda: exponenciales, secuenciales y aleatorias. Dentro de las primeras se encuentran aquellas estrategias que tienen complejidad $O(2^n)$ pero aseguran la obtención del subconjunto óptimo. Las estrategias secuenciales a diferencia de las exponenciales recorren sólo una porción del espacio de búsqueda y por tanto no aseguran la obtención del óptimo, aunque el coste computacional es polinomial. Las estrategias aleatorias se basan en visitar diferentes regiones del espacio de búsqueda sin un orden predefinido, evitando de esta forma que se pueda obtener un óptimo local de la función de evaluación para un determinado subconjunto de atributos.

4.2.2. Generación de subconjuntos

Todo proceso de selección de atributos tiene un punto de partida, que puede ser el conjunto completo de atributos, el conjunto vacío o cualquier estado intermedio. Tras evaluar el primer subconjunto, se examinarán otros subconjuntos generados según una dirección de búsqueda (hacia adelante, hacia atrás, aleatoria o cualquier variación o mezcla de las anteriores). El proceso terminará cuando recorra todo el espacio o cuando se cumpla una condición de parada, según la estrategia de búsqueda seguida.

Dirección de búsqueda

Se entiende por dirección de búsqueda, la relación entre los atributos de un subconjunto con el siguiente, al realizar el recorrido a través del espacio de búsqueda. Se puede seguir uno de los siguientes esquemas [23]:

Secuencial hacia adelante (forward): Este esquema empieza con el conjunto vacío y se añaden secuencialmente atributos del conjunto original. En cada iteración, se elige el mejor atributo entre los no seleccionados, basándose en alguna función de evaluación. El algoritmo puede parar cuando llegue a un número determinado de atributos seleccionados, o cuando la evaluación sobrepase un valor prefijado, o simplemente, al dejar de aumentar la evaluación. Si se deja ejecutar el algoritmo sin un criterio de parada, se obtiene un ranking de atributos, es decir, una lista de atributos ordenada según cuándo se incorpore cada atributo al subconjunto final.

El principal inconveniente de hacer un recorrido siguiendo esta dirección, es la posibilidad de no detectar interacciones básicas entre atributos que sean muy interesantes para la clasificación. Es decir, puede ocurrir que atributos que por separado son irrelevantes, el estar juntos en un determinado subconjunto les haga ser muy importantes.

Secuencial hacia atrás (backward): Se comienza con el conjunto completo de atributos y se eliminan secuencialmente atributos. En cada iteración, basándose en alguna función de evaluación, se escoge el atributo menos relevante de entre los originales, eliminándolo del conjunto.

Al igual que en el caso anterior, el algoritmo puede parar cuando llegue a un número determinado de atributos en el subconjunto, o cuando la evaluación alcance cierto umbral predefinido, o simplemente, al dejar de aumentar la evaluación. Igualmente, se puede obtener un ranking de atributos si se espera a llegar hasta el final, aunque no se aconseja por ir ordenado según la irrelevancia de los atributos.

En este caso, se remedia en parte el inconveniente de la generación hacia adelante, aunque permanezcan interacciones ocultas. Se puede generalizar para que devuelva subconjuntos de k elementos, aunque el costo se eleva.

Aleatoria: No se conoce el punto de inicio, ni cual es el siguiente subconjunto generado. Se pasa de un subconjunto a otro sin ningún orden establecido, añadiendo o eliminando uno o varios atributos. De esta forma se pretende evitar elegir subconjuntos de atributos que sean óptimos locales, como puede ocurrir en los casos anteriores. El criterio de parada suele ser la ejecución de un número fijo de iteraciones.

Otros esquemas se pueden obtener variando o mezclando algunos de los anteriores, como el *Bidireccional*, que realiza la búsqueda hacia adelante y hacia atrás al mismo tiempo, o el esquema *Compuesto*, que aplica una serie de pasos consecutivos hacia adelante y otra serie de pasos consecutivos hacia atrás.

Estrategia de búsqueda

Para una base de datos con *n* atributos, existen 2^n subconjuntos candidatos. Una búsqueda exhaustiva en este espacio es totalmente ineficiente, incluso para bases de datos pequeñas, siendo necesario el uso de diferentes estrategias para atajar este problema. A continuación, se explican brevemente las características más importantes de los tipos de estrategias de búsqueda según [23]: completa, secuencial y aleatoria.

Completa: Esta búsqueda garantiza la localización de un resultado óptimo conforme a un criterio dado. Si para seleccionar los subconjuntos óptimos se ha de examinar todos los conjuntos posibles del espacio, coincidirá con la exhaustiva. Sin embargo, según la medida de evaluación utilizada, puede no ser necesario examinar todos los subconjuntos posibles.

Es posible afirmar que una búsqueda exhaustiva siempre es completa, pero a la inversa no se cumple en todos los casos [33]. Algunas implementaciones de búsqueda completa pero no exhaustiva son "*Branch & Bound*" [30] y "*Beam*" [10].

Dos implementaciones clásicas de la búsqueda exhaustiva son la búsqueda en profundidad y en anchura. Ambos métodos se pueden encontrar con generación de subconjuntos hacia adelante y hacia atrás. Representando el espacio de búsqueda mediante un árbol, es decir, se empieza con un atributo X_1, y se sigue con las combinaciones de dos atributos donde intervenga el anterior, y a continuación las combinaciones de tres donde aparezca X_1, y así sucesivamente hasta el conjunto total de atributos. A continuación, el mismo proceso se realiza con el resto de atributos sin repetir ningún conjunto visitado anteriormente.

Por otro lado, el recorrido en anchura va examinando el árbol por niveles, es decir, primero cada atributo individualmente, luego todas las combinaciones de dos, a continuación de tres, y así hasta llegar al conjunto completo.

Secuencial: Son estrategias que realizan una búsqueda parcial a través del espacio formado por los atributos, con el riesgo de no encontrar subconjuntos óptimos. Lógicamente son algoritmos más rápidos que los pertenecientes al grupo anterior. En este grupo se encuentran distintas variantes de la técnica Greedy, que añaden o eliminan atributos uno a uno [22]. Otra alternativa es añadir (o eliminar) p atributos en un paso y eliminar (o añadir) q atributos en el siguiente (p > q) [10]. Los algoritmos con búsqueda secuencial son simples de implementar y rápidos en generar resultados, generalmente, el orden del espacio de búsqueda es $O(n^2)$ o menor.

Aleatoria: Las dos estrategias anteriores son determinísticas, es decir, todas las veces que se ejecutan devuelven el mismo resultado. En este apartado se introducen una serie de algoritmos que tienen la propiedad contraria, no determinísticas, es decir, no se espera que en diferentes ejecuciones proporcionen exactamente el mismo resultado. Este tipo de estrategias, al contrario que las anteriores, busca a través del espacio formado por los atributos de manera aleatoria, es decir, no existe un estado siguiente o anterior que se pueda determinar según alguna regla. En este tipo de búsqueda se pretende usar el carácter aleatorio para no caer en mínimos locales e incluso moverse temporalmente a otros estados con peores soluciones. Además, pueden detectar interdependencias entre atributos que la búsqueda heurística no captaría. Estos algoritmos se pueden detener en cualquier momento [48], por lo que es difícil determinar su costo. Normalmente, el criterio de parada es un número de

iteraciones, y no se puede tener garantías de que el subconjunto elegido sea el óptimo. Además de la implementación genérica de un algoritmo aleatorio, se encuentran en este grupo las búsquedas basadas en algoritmos evolutivos, búsqueda Tabú y “Simulated Annealing” para selección de atributos. En [37] se aplican redes neuronales en la selección de atributos. Para ello, se entrena la red y posteriormente se poda las conexiones entre capas tanto como sea posible sin sacrificar la predicción. Se eliminan las unidades de entrada sin conexión con las unidades ocultas, y se selecciona el resto de unidades.

Capítulo 5

Support Vector Machines

En este capítulo se describe la técnica de Support Vector Machines para clasificación, mostrando su formulación mediante un problema de minimización cuadrático con un número de variables que es igual al número de objetos del entrenamiento.

La tarea de entrenamiento de los SVM involucra optimización de una función convexa, por ende no hay mínimos locales que “compliquen” el proceso de aprendizaje. El enfoque tiene muchos otros beneficios, por ejemplo, el modelo construido tiene una dependencia explícita en los patrones de mayor aporte en los datos (los Support Vectors). Además, permite generalizar de mejor forma frente a nuevos objetos [41] dado que considera el principio de “minimización del riesgo estructural”.

La clasificación mediante Support Vector Machines permite obtener clasificadores lineales y no lineales. Para ello se estudiará el caso de objetos linealmente separables, luego se verá el caso linealmente no separable para finalmente revisar la clasificación no lineal.

5.1 Minimización del riesgo estructural (SRM)

El Riesgo Estructural [41] nace como necesidad de incorporar la capacidad de generalización de manera explicita en la construcción de un modelo predictivo y prevenir, de esta forma, el problema de sobreajuste.

Se define la capacidad de una máquina de aprendizaje como la habilidad para aprender cualquier conjunto de entrenamiento sin error [41]. En el siguiente ejemplo se observa que si se busca clasificar una muestra de árboles de acuerdo a si son de color verde o no, el poder discriminatorio de esta función es muy bajo, asumiendo que casi todos los árboles son verdes y por ende esta clasificación no logra hacer una diferencia importante. Por otro lado, si se clasifica de acuerdo al número de hojas, es probable que ningún árbol tenga el mismo número que otro, por lo tanto la función de clasificación se está ajustando demasiado a los datos y perdiendo su capacidad de generalización. La Figura 5.1 esquematiza este fenómeno, donde en el diamante se define la clasificación usada en los datos, la cual genera dos o más subconjuntos de acuerdo a si cumplen o no la regla de clasificación, simbolizados por el árbol sin cruz y con cruz respectivamente.

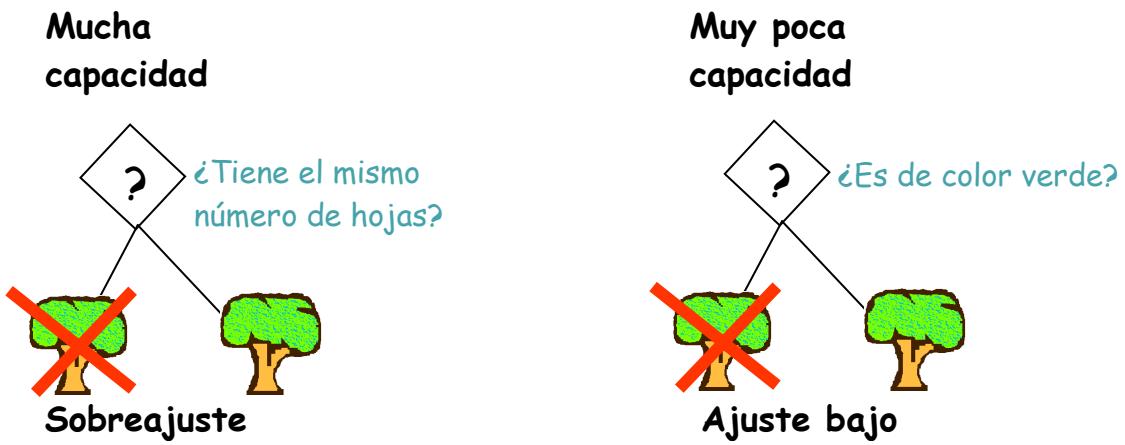

Figura 5.1 Ejemplo Generalización

Para solucionar este problema, los métodos que aplican este criterio (y en particular Support Vector Machines) buscan el balance correcto entre precisión y capacidad de generalización. Para ello se restringe la complejidad de la función de clasificación. Intuitivamente una función simple que explique más de los datos es preferible a una más compleja. En [41] Vapnik afirma y demuestra que maximizando el margen de separación entre clases es posible generar funciones de clasificación que logren este balance.

5.2 Descripción del modelo

La técnica de Support Vector Machines fue propuesta por Vapnik [39,40]. Ésta se basa en encontrar un hiperplano de separación que divida el espacio de entrada en dos regiones. Cada una de estas regiones corresponderá a una de las clases definidas, como se muestra en la Figura 5.2:

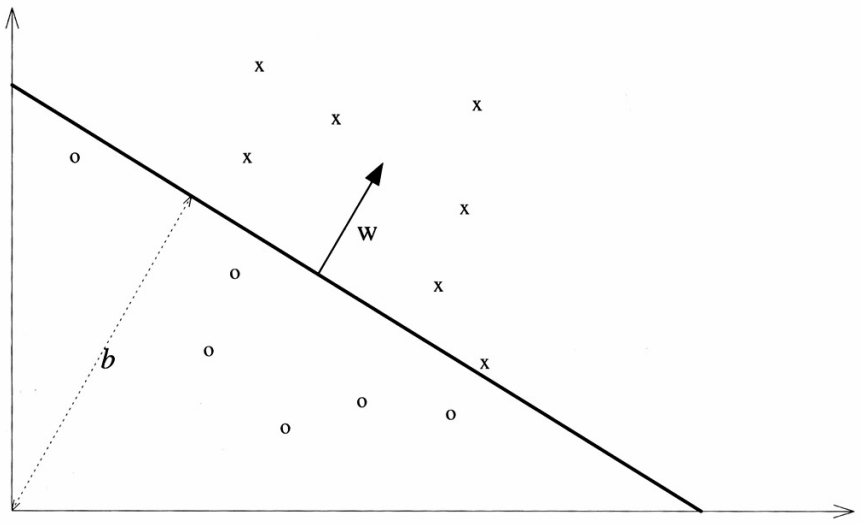

Figura 5.2 Un Hiperplano de clasificación { $\vec{w}$, b } para un conjunto de entrenamiento de dos dimensiones

Por un lado, a hiperplanos que están más alejados de las fronteras de las clases de objetos corresponden mayores márgenes de separación. Por otro, hiperplanos que aciertan más en la asignación de objetos a las clases a las que efectivamente pertenecen, tienen un menor error de clasificación. Por lo tanto, un hiperplano de separación ideal debe maximizar el margen de separación y minimizar el error de clasificación. Sin embargo, no siempre es posible cumplir los dos objetivos simultáneamente. Para salvar esta dificultad se plantea un problema de optimización cuya función objetivo combina ambos propósitos.

Este problema de optimización resulta ser un problema de minimización cuadrático convexo [8]. En el caso que el número de objetos a clasificar es mayor que el número de atributos de cada objeto, lo que usualmente sucede,

este problema tiene una única solución óptima. Lo descrito anteriormente corresponde al caso que existe un hiperplano de separación de las clases. En ese caso se dice que las clases son linealmente separables. Recientemente, el estudio de los SVM se ha extendido al caso de clases que no son linealmente separables mediante la introducción de las llamadas funciones Kernel [31,36] o de variables de pérdida u holgura [7]. Esta metodología ha sido también extendida para problemas de regresión [4].

Se considera un problema de clasificación binaria para el cual ya se ha definido el conjunto de entrenamiento. Para el caso de Credit Scoring, los objetos a clasificar son los clientes. Para cada cliente se han definido *n* variables (o atributos) a estudiar y que se tiene m clientes en el conjunto de entrenamiento. Entonces, cada cliente es representado por un vector característico de dimensión *n + 1* cuyas primeras *n* coordenadas corresponden a las variables del estudio y la última corresponde a la clase a la que el cliente pertenece.

Se denotan estos vectores como un par $(\vec{x}, y)$ donde $\vec{x} = (x^1, ..., x^n)$ son las variables del estudio y la última coordenada $y \in \{-1, +1\}$ indica a qué clase pertenece el cliente. En particular, se denota $(\vec{x_i}, y_i)$ al vector característico correspondiente al cliente *i* (para *i = 1,. . .,m*). En este sentido, cuando quede claro del contexto, si el cliente *i* es representado por un vector de la forma $(\vec{x_i}, +1)$ se dirá que ese cliente está en la clase positiva o bien, que el vector $\vec{x_i}$ está en la clase positiva. De manera análoga, para un cliente *i*, representado por un vector de la forma $(\vec{x_i}, -1)$, se dirá que él o su vector $\vec{x_i}$ están en la clase negativa.

5.3 Caso linealmente separable

Un supuesto inicial para este trabajo es que el conjunto de entrenamiento es linealmente separable. Es decir, existe un hiperplano en $\Re^n$ que deja a todos los vectores $\vec{x_i}$ asociados a los clientes de una clase de un lado del hiperplano y al los de la otra del otro lado. Formalmente, existe un par $(\vec{w}, b) \in \Re^{n+1}$ tal que $\vec{w} \cdot \vec{x_i} + b > 0$ si el cliente *i* está en la clase positiva y $\vec{w} \cdot \vec{x_i} + b < 0$, si está en la clase negativa.

De esta manera, dado un vector no nulo normal al hiperplano de separación $\vec{\alpha} \in \Re^n$, se define una función de clasificación $f : \Re^n \to \Re$ por la expresión $f(\vec{x}) \equiv \vec{w} \cdot \vec{x} + b$. Entonces si $f(\vec{x}_i) > 0$, el cliente *i* está en la clase positiva y si $f(\vec{x}_i) < 0$, *i* está en la clase negativa.

Para definir el margen de clasificación, se consideran las distancias d^+ y d^-. La distancia d^+ es la distancia euclideana entre el hiperplano y la clase positiva. Es decir, la distancia entre el hiperplano y el punto en la clase positiva más cercano a éste. De manera análoga, pero respecto de la clase negativa se define d^-. Una vez determinadas estas distancias, se define el margen de separación del hiperplano como la suma $d^+ + d^-$.

Para clasificar correctamente a todos los clientes en el conjunto de entrenamiento debemos imponer las siguientes restricciones:

$$\begin{aligned} &\vec{w} \cdot \vec{x}_i + b > 0 \quad \text{para los clientes i en la clase positiva y} \\ &\vec{w} \cdot \vec{x}_i + b < 0 \quad \text{para los clientes i en la clase negativa.} \end{aligned} \tag{5.1}$$

No es una buena práctica incluir desigualdades estrictas como restricciones en problemas de optimización. Por lo tanto, sería importante poder encontrar condiciones equivalentes a las anteriores pero que sean del tipo “menor o igual” o “mayor o igual”.

Recordando que si un hiperplano está definido por el par $(\vec{w}, b)$, entonces cualquier par de la forma $(\lambda\vec{w}, \lambda b)$ con $\lambda > 0$ también define el mismo hiperplano, y si $\vec{w} \cdot \vec{x} + b > 0$ para un par particular $(\vec{w}, b)$, entonces existe un par $(\vec{w_i}, b_i)$ que define el mismo hiperplano y tal que $\vec{w} \cdot \vec{x_i} + b_i \geq 1$.

Para la clase negativa, se puede razonar de manera análoga. Como el conjunto de entrenamiento es finito, si existe un par $(\vec{w}, b)$ que satisface las restricciones para todos los clientes del conjunto de entrenamiento, entonces existe un par $(\vec{w}', b')$ que define el mismo hiperplano tal que el lado izquierdo de las

restricciones tiene valor absoluto mayor o igual a 1. De esta manera, las condiciones planteadas pueden ser incluidas en el problema de optimización en la forma:

$$\begin{aligned} &\vec{w}\cdot\vec{x}_i + b \geq 1 \quad \text{para los clientes i en la clase positiva y} \\ &\vec{w}\cdot\vec{x}_i + b \leq -1 \quad \text{para los clientes i en la clase negativa.} \end{aligned} \tag{5.2}$$

Las observaciones (clientes para el caso de Credit Scoring) para las cuales una de estas restricciones es activa, es decir $|\vec{w}\cdot\vec{x}_i + b| = 1$ tienen especial importancia. En particular, estos puntos yacen exactamente en los llamados hiperplanos canónicos que están definidos por las ecuaciones [9]:

$$\begin{aligned} \vec{w}\cdot\vec{x} + b &= +1 \\ \vec{w}\cdot\vec{x} + b &= -1 \end{aligned} \tag{5.3}$$

Se denotará por x^+ a una observación cualquiera que esté en el hiperplano canónico de la clase positiva (lado derecho igual a +1) y por x^- a un cliente cualquiera que esté en el hiperplano canónico de la clase negativa. El hiperplano definido por el par $(\vec{w}, b)$ es paralelo a los hiperplanos canónicos. Con esta notación, el margen de separación del hiperplano definido por el par $(\vec{w}, b)$, y denotado por γ, puede ser expresado como [9]:

$$\gamma = \frac{1}{2}\left[\left(\frac{\vec{w}\cdot x^+}{\|\vec{w}\|}\right) - \left(\frac{\vec{w}\cdot x^-}{\|\vec{w}\|}\right)\right] \tag{5.4}$$

Manipulando algebraicamente la expresión anterior se obtiene:

$$\gamma = \frac{1}{2\|\vec{w}\|}\left[\left(\vec{w}\cdot x^+\right) - \left(\vec{w}\cdot x^-\right)\right] = \frac{1}{\|\vec{w}\|} \tag{5.5}$$

Se tiene entonces que la distancia entre el hiperplano definido por el par $(\vec{w},b)$ a cada uno de los hiperplanos canónicos es igual a $1/\|\vec{w}\|$. Además, los puntos

más cercanos a este hiperplano en el conjunto de entrenamiento están en los hiperplanos canónicos. Por lo tanto, el margen de separación de este hiperplano es igual a $2/\|\vec{w}\|$. En la Figura 5.3 se ilustra esta situación: *H1* y *H2* son los hiperplanos canónicos.

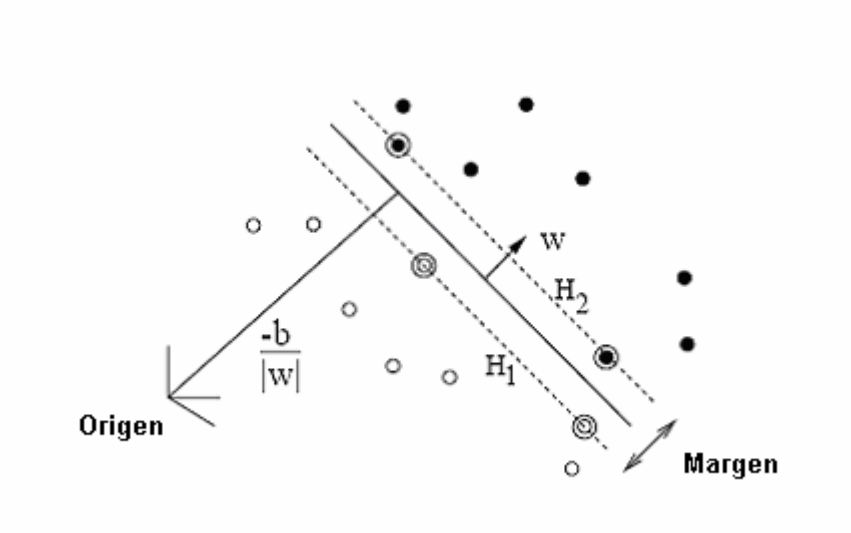

Figura 5.3 Caso linealmente separable

Se desea maximizar este margen y esto es equivalente a minimizar la norma euclidiana de $\vec{w}$. De esta manera el problema de optimización que vamos a resolver es el siguiente [9]:

$$\textbf{(P)} \qquad \min_{\vec{w},b} \frac{\|\vec{w}\|^2}{2} \qquad \textbf{(5.6)}$$

s.a.

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0 \qquad \text{para i = 1,...,m.}$$

Resolviendo este problema se obtiene un vector $\vec{w}^*$ óptimo que permite calcular el margen geométrico máximo de un hiperplano de separación, $\gamma^* = 2/\|\vec{w}^*\|$.

Resolución Analítica del Problema de Optimización

El problema de optimización planteado anteriormente es un problema cuadrático convexo y tiene una única solución óptima global. Esta solución puede ser obtenida encontrando una solución para las condiciones de optimalidad de primer orden, conocidas como las condiciones de Karush, Kuhn y Tucker [9]. Para esto definimos el Lagrangiano de **(P)**,

$$Lp = \frac{|| \vec{w} ||^2}{2} - \sum_{i=1}^{m} \alpha_i y_i (\vec{x}_i \cdot \vec{w} + b) + \sum_{i=1}^{m} \alpha_i \qquad \textbf{(5.7)}$$

donde los α_i son los multiplicadores de Lagrange asociados a las restricciones de **(P)**. A partir de este Lagrangiano podemos plantear las condiciones de optimalidad (Condiciones de Karush – Kuhn - Tucker) [2]:

$$\frac{dLp}{dw_j} = w_j - \sum_{i=1}^{m} \alpha_i y_i x_i^j = 0 \qquad \text{i = 1,...,n.} \qquad \textbf{(5.8)}$$

$$\frac{dLp}{db} = -\sum_{i=1}^{m} \alpha_i y_i = 0 \qquad \textbf{(5.9)}$$

$$y_i(\vec{x_i} \cdot \vec{w} + b) - 1 \geq 0 \qquad \text{i = 1,...,m.} \qquad \textbf{(5.10)}$$

$$\alpha_i \geq 0 \qquad \text{i = 1,...,m.} \qquad \textbf{(5.11)}$$

$$\alpha_i (y_i(\vec{x_i} \cdot \vec{w} + b) - 1) = 0 \qquad \text{i = 1,...,m.} \qquad \textbf{(5.12)}$$

De las condiciones **(5.8)** y **(5.9)** se obtienen, en el óptimo, las siguientes relaciones:

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x_i} \tag{5.13}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{5.14}$$

Al reemplazar estas expresiones en la ecuación **(5.7)** se puede llevar el Lagrangiano a la forma:

$$L_D = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s \vec{x_i} \cdot \vec{x_s} \tag{5.15}$$

A partir de esta expresión, aplicando dualidad lagrangiana se puede obtener un problema dual conocido como el Dual de Wolfe [2,13] de **(P)**:

(D) $$\max_{\alpha} L_D \equiv \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s \vec{x_i} \cdot \vec{x_s} \tag{5.16}$$

s.a

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$\alpha_i \geq 0$ para i = 1,...,m.

En el caso de problemas de optimización convexos regulares, como el problema **(P)**, el Dual de Wolfe **(D)** es un dual fuerte. La resolución se reduce, entonces, a obtener los valores óptimos para los multiplicadores α_i. Una vez conocidos éstos, los valores óptimos de las variables primales $\vec{w}$ se obtienen de las ecuaciones **(5.13)** y **(5.14).** Se define la función discriminante como:

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b = \sum_{i=1}^{m} \alpha_i y_i (\vec{x_i} \cdot \vec{x}) + b \tag{5.17}$$

Resumiendo, para encontrar el par $(\vec{w}^*, b^*)$ que define el hiperplano de separación óptimo se resuelve el problema dual **(D)** obteniéndose los multiplicadores α^* óptimos.

5.4 Caso linealmente no separable

Al aplicar el algoritmo anterior en un caso linealmente no separable, no se llegará a una solución factible ya que no existirá un hiperplano que cumpla con la restricción **(5.10)**. Para extender estas ideas al caso linealmente no separable se desea relajar esta restricción pero sólo cuando sea necesario, esto es, se introduce un costo extra (i.e. un incremento en la función objetivo del Primal) por hacer esto. Esto puede hacerse introduciendo variables de holgura positivas ξ_i en las restricciones. Las nuevas restricciones quedan de la siguiente manera:

$$y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1 - \xi_i \qquad \textbf{(5.18)}$$

$$\xi_i \geq 0 \quad \forall i \qquad \textbf{(5.19)}$$

Luego, si un error ocurre, el correspondiente ξ_i debe exceder la unidad. De acuerdo a esto $\sum_i \xi_i$ es cota superior del total de errores en el entrenamiento. Una manera natural de incorporar este costo extra en la función objetivo es cambiarla de $\min_{\vec{w},b} \frac{\|\vec{w}\|^2}{2}$ a $\min_{\vec{w},b,\xi} \frac{\|\vec{w}\|^2}{2} + C(\sum_i \xi_i)^k$, donde C es un parámetro definido por el investigador. Un alto valor para C corresponde a asignar un alto costo de penalización por errores. El problema anterior es convexo para cualquier *k* entero positivo; para *k=2* y *k=1* es un problema de programación cuadrática. La elección de *k=1* tiene una ventaja adicional ya que ni los ξ_i ni sus multiplicadores de Lagrange asociados aparecen en el problema Dual de Wolfe,

conservando su forma **(D)**, y la única diferencia es que ahora para el hiperplano óptimo los α_i están ahora acotados superiormente por C [41].

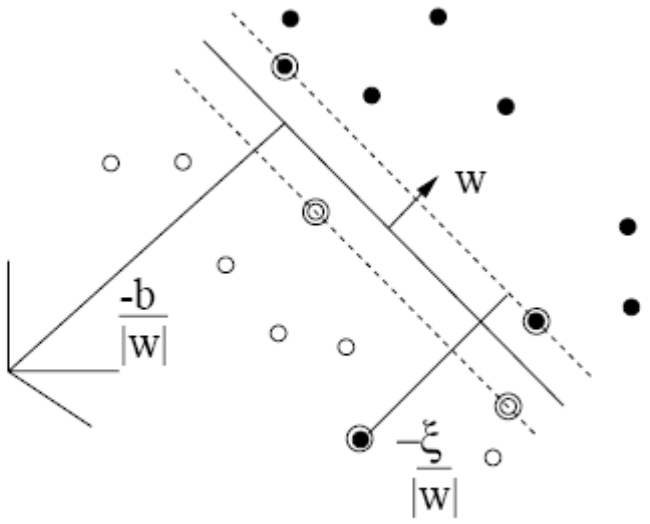

Figura 5.4 Caso linealmente no separable

5.5 SVM no lineal

¿Cómo es posible generalizar los métodos anteriores cuando la función de decisión no es lineal en los datos? Un antiguo truco propuesto por Aizerman en 1964 puede ser usado para conseguir esta generalización [41].

Primero, es importante notar que los datos sólo aparecen en forma de productos punto $\vec{x}_i \cdot \vec{x}_s$. La idea es proyectar los objetos en otro espacio euclidiano (también llamado espacio de Hilbert) *H* de mayor dimensión (incluso infinita) en el cuál sean linealmente separables, luego encontrar el hiperplano en ese nuevo espacio para finalmente retornar al espacio original tanto los objetos como el hiperplano. Éste último ya no será "hiperplano" en el espacio original sino una "hipersuperficie" no lineal.

La proyección de los objetos se debe realizar mediante una función no lineal Φ:

$$\Phi : \Re^n \rightarrow H \tag{5.20}$$

Luego el algoritmo de entrenamiento depende de los datos sólo a través de los productos punto en H, i.e. en funciones de la forma $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_s)$. Ahora, si existe una "función Kernel" *K* tal que $K(\vec{x}_i, \vec{x}_s) = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_s)$ [42], sólo sería necesario usar *K* en el entrenamiento, y no se requeriría explicitar Φ.

La elección de una función Kernel apropiada para un problema en particular es aún motivo de investigación.

¿Para qué función Kernel existe un par $\{H, \Phi\}$ que cumpla las condiciones mencionadas anteriormente? La respuesta es la condición de Mercer [40]: Existe una proyección Φ y una expansión

$$K(x, y) = \sum_i \Phi(x)_i \cdot \Phi(y)_i \tag{5.21}$$

si y sólo si, para cualquier $g(x)$ tal que

$$\int g(x)^2 dx \tag{5.22}$$

es finita, entonces

$$\int K(x, y) g(x) g(y) dx dy \geq 0 \tag{5.23}$$

Esto permite asegurar que el Hessiano de la formulación Dual esté definido y el problema cuadrático tenga solución [4].

Entre los distintos tipos de funciones Kernel que cumplen está condición están:

(a) Polinomio de grado d: $K(\vec{x}_i, \vec{x}_s) = \left(\vec{x}_i \cdot \vec{x}_s + 1\right)^d$, $d \in \mathbb{N}$

(b) Radial Basis Function (RBF)[1]: $K(\vec{x}_i, \vec{x}_s) = \exp\left(-\frac{\| \vec{x}_i - \vec{x}_s \|^2}{2\rho^2} \right)$, $\rho > 0$

(c) Función de Transferencia Tangencial[2]: $K(\vec{x}_i, \vec{x}_s) = \tanh(\vec{x}_i \cdot \vec{x}_s - \theta)$

Para realizar la creación del modelo se reemplaza el producto escalar entre los vectores $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_s)$ en la formulación **(D)**, quedando el siguiente problema a resolver:

$$\max_{\alpha} L_D \equiv \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s K(\vec{x}_i, \vec{x}_s) \qquad \textbf{(5.24)}$$

s.a

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \qquad \text{para } i = 1,\ldots,m.$$

La función discriminante del caso lineal **(D)** se modifica quedando de la siguiente forma:

$$f(\vec{x}) = \sum_{i \in SV} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b \qquad \textbf{(5.25)}$$

Notar que los objetos que intervienen en esta representación son aquellos que son Support Vector ($\vec{x}_i$ tal que $i \in SV$). Recordar que un objeto es Support Vector si el multiplicador asociado (α_i) es mayor estricto a cero y menor e igual a C.

Se considera el siguiente ejemplo a modo explicativo (Figuras 5.5, 5.6):
Se dispone de los siguientes datos en $\Re^2$:

[1] También conocido como Kernel Gaussiano
[2] Utilizada comúnmente en Redes Neuronales para MLP, para algunos valores de θ

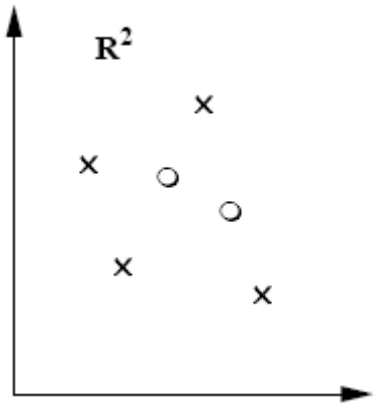

Figura 5.5 Ejemplo SVM no lineal(1)

El conjunto no es linealmente separable pero es fácil encontrar un espacio H en el cual lo sean. Se elige $H = \Re^3$, el Kernel $K(\vec{x}_i, \vec{x}_s) = (\vec{x}_i \cdot \vec{x}_s)^2$ y

$$\Phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \tag{5.26}$$

Utilizando la proyección en $\Re^3$ se encuentra una función lineal que separa los datos correctamente **(a)**. Finalmente se aplica la transformación inversa para llevar, tanto los datos como la función de clasificación, al espacio original ($\Re^2$), donde esta función deja de ser lineal **(b)**.

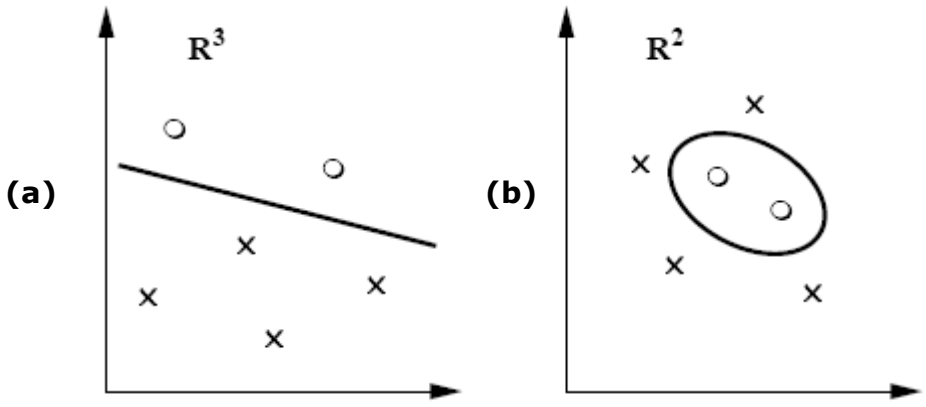

Figura 5.6 Ejemplo SVM no lineal(2)

5.6 Características de los Support Vector Machines

Esta herramienta presenta notorias ventajas en comparación con otros algoritmos como las Redes Neuronales en los siguientes aspectos [40,41]:

- Combina aspectos geométricos fáciles de visualizar y elementos teóricos de importante fundamento, como es Dualidad, condiciones de KKT, funciones Kernel y el Riesgo Estructural.

- Permite la construcción de separadores no lineales mediante funciones de Kernel gracias a la formulación Dual.

- La función discriminante depende sólo de una cantidad reducida de objetos (los Support Vectors) que están más cerca de la superficie clasificadora, a diferencia de métodos estadísticos como el Análisis Discriminante, en el cual valores extremos determinan el hiperplano construido. De acuerdo a esto, la velocidad y exactitud del modelo depende sólo de un número reducido de los objetos originales.

- Mediante las funciones Kernel se puede obtener el valor de producto punto entre las proyecciones de los objetos. De esta forma, no es necesario realizar explícitamente la proyección de éstos.

- Tiene un desempeño computacional más eficiente que otros algoritmos de Inteligencia Artificial (Redes Neuronales por ejemplo).

- Minimiza el problema de sobreajuste (overfitting), debido a que se preocupa de no obtener modelos muy ajustados a los datos, mediante la maximización del margen entre los hiperplanos paralelos al hiperplano separador. Para esto, minimiza la norma del vector normal al hiperplano separador.

- La solución obtenida es un mínimo global debido a la convexidad de la función objetivo y de la región factible.

Sin embargo, también posee algunas limitaciones:

- Trabaja con datos numéricos, y por lo tanto necesita una codificación especial para emplear datos originalmente nominales.

- Para obtener la clasificación no lineal se necesita proyectar o “mapear” los objetos en un espacio de mayor dimensión. La representación de los productos punto mediante la función Kernel requiere de múltiples cálculos. Esta matriz puede tornarse singular al tener muchos atributos poco relevantes o con valores muy pequeños, por lo que la convergencia de este problema se ve fuertemente afectada.

- La utilización de las funciones Kernel puede determinar la solución obtenida. En la actualidad no se cuenta con un consenso acerca de qué función utilizar para cada tipo de problema, lo que se deja al “ensayo y error” la elección del tipo de función adecuada.

Capítulo 6

Selección de Atributos vía Support Vector Machines

6.1 Motivación

Si bien SVM ha mostrado buenos resultados en cuanto a clasificación [19] no ha sucedido lo mismo en cuanto a la selección de atributos [5,38]. Esto se debe principalmente a que no fueron diseñados para este propósito y porque al utilizar la norma-2 para el cálculo del margen en la función objetivo no anula componentes en el vector normal al hiperplano separador $\vec{w}$.

El componente de la función objetivo que permitiría realizar la selección de atributos es $\|\vec{w}\|^2$, ya que al minimizar la norma de este vector, se minimiza cada uno de sus componentes. Cada componente del vector $\vec{w}$ multiplica a los atributos de los objetos en la función discriminante lineal. Esto implica que en esta formulación cada componente de $\vec{w}$ puede representar a cada uno de los atributos originales. Por ejemplo, si el valor del coeficiente j de $\vec{w}$ es muy grande en magnitud se puede concluir que el atributo j es muy importante para discriminar, considerando que los atributos tienen la misma escala[3]. En cambio, si el componente w_j toma el valor cero, indica que el atributo no es considerado al momento de realizar la clasificación.

Se considera el siguiente ejemplo:

[3] Normalmente los valores de los atributos son escalados entre 0 y 1, o normalizados con media 0 y desviación estándar igual a 1

Sean $\overrightarrow{w_1} = (0.5, 0.5, 0.5, 0.5)$ y $\overrightarrow{w_2} = (1, 0, 0, 0)$. Luego $\| \overrightarrow{w_1} \|^2 = \| \overrightarrow{w_2} \|^2 = 1$, pero en el primer caso se utilizan 4 atributos, donde cada componente de $\vec{w}$ corresponde a un atributo del problema, mientras que en el segundo caso solamente se necesita el primer atributo. Los atributos cuyo componente en $\vec{w}$ son nulos, no son utilizados en la clasificación, por lo que pueden ser eliminados del modelo.

6.2 Trabajos anteriores

SVM Penalizado para Selección de Atributos [28]

La intención de la selección de atributos es forzar a una solución que utilice la mínima cantidad de atributos originales (anulando componentes de $\vec{w}$) considerando que se puede encontrar una solución peor que utilizando los atributos, lo que debe ser determinado por el investigador.

En [28] se propone una modificación en la formulación del problema a resolver mediante Support Vector Machines de modo de incorporar en la función objetivo una penalización por cada atributo utilizado para realizar la clasificación. Este enfoque de selección de atributos se describe sólo para el caso lineal.

Para mostrar la nueva formulación se introduce la siguiente notación:

Sea

$$\vec{w} \in \Re^m : \quad |\vec{w}|_j = \begin{cases} w_j & si \ \ x_j > 0 \\ 0 & si \ \ x_j = 0 \\ -w_j & si \ \ x_j < 0 \end{cases} \qquad j = 1, ..., m \tag{6.1}$$

Función de Paso:

$$\left(\vec{w}^*\right)_j = \begin{cases} 1 & si \ \ x_j > 0 \\ 0 & si \ \ x_j = 0 \\ -1 & si \ \ x_j < 0 \end{cases} \qquad j = 1, ..., m \tag{6.2}$$

Notar que si $\vec{w} \in \Re_+^m$, $\left(\vec{w}*\right)_j \in \{0,1\}, j = 1,...,m$.

Número de componentes estrictamente positivos de $\vec{x}$:

$$w_j \in \{0,1\} \forall j \Rightarrow \vec{e}^T \cdot \vec{w} = \sum_{j=1}^{m} w_j \qquad \textbf{(6.3)}$$

Se propone la siguiente función objetivo:

$$\frac{||\vec{w}||^2}{2} + C_1 \sum\nolimits_i \xi_i + C_2 \vec{e}^T \cdot |\vec{w}|_* \qquad \textbf{(6.4)}$$

donde $\frac{||\vec{w}||^2}{2} + C_1 \sum_i \xi_i$ corresponde a la formulación tradicional de la función objetivo para SVM linealmente no separable y $C_2 \vec{e}^T \cdot |\vec{w}|_*$ es la suma de los componentes de $\vec{w}$ no negativos y se refiere al número de atributos seleccionados (cardinalidad de $\vec{w}$, lo que es equivalente a la norma 0 de $\vec{w}$ propuesta en [5]).

La formulación anterior presenta la inconveniencia de no ser una función continua dado que incorpora las funciones módulo y de paso, que son discontinuas.

Reemplazando el módulo por variables auxiliares y la función de paso por una aproximación exponencial cóncava [5,28] se obtiene el siguiente modelo:

$$\min_{\vec{w},\vec{v},\xi_i,b} \frac{||\vec{w}||^2}{2} + C_1 \sum\nolimits_i \xi_i + C_2 \vec{e}^t \cdot \left(\vec{e} - \exp(-\tau \cdot \vec{v})\right) \qquad \textbf{(6.5)}$$

s.a.

$$y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1 - \xi_i \qquad \forall i$$
$$\xi_i \geq 0 \qquad \forall i$$
$$-\vec{v} \leq \vec{w} \leq \vec{v}$$

Donde $\vec{w} \in \Re^m$, $\vec{v} \in \Re_+^m$ y $\xi_i, b, \tau \in \Re$. Este modelo fue llamado LP-SVM. Para encontrar los mejores parámetros fue usada la técnica Validación Cruzada (Cross Validation) [28]

Selección de genes para clasificación del cáncer utilizando Support Vector Machines [16]

La siguiente proposición busca incorporar la selección de atributos en la construcción de la función de clasificación para el caso no lineal.

Recordando el procedimiento para derivar la formulación no lineal a través de la formulación Dual del problema original, se tiene el siguiente problema de optimización:

$$\max_{\alpha} L_D \equiv \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s K(\vec{x_i}, \vec{x_s}) \qquad \textbf{(6.6)}$$

s.a

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \qquad \text{para } i = 1,\ldots,m.$$

donde $K(\vec{x_i}, \vec{x_s}) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_s})$. Se introduce el producto por componentes entre dos vectores:

$$\vec{x} * \vec{w} = (x_1 w_1, \ldots, x_n w_n) \qquad \textbf{(6.7)}$$

Es posible introducir la selección de atributos en la función de Kernel de la siguiente manera:

$$K_{\vec{\sigma}}(\vec{x_i}, \vec{x_s}) \equiv K(\vec{\sigma} * \vec{x_i}, \vec{\sigma} * \vec{x_s}) \qquad \textbf{(6.8)}$$

donde σ_j es la variable auxiliar que representa la selección del atributo *j* ($\sigma_j = 1$ si se selecciona este atributo y 0 en caso contrario). Considerando la siguiente formulación del problema dual es posible manejar como parámetros los atributos a utilizar (se resuelve usando el algoritmo RFE-SVM descrito en la siguiente página):

$$\max_{\alpha,\sigma} L_D \equiv \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s K_{\sigma}(\vec{x}_i, \vec{x}_s) \qquad \textbf{(6.9)}$$

s.a

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \qquad \text{para } i = 1,\ldots,m.$$

$$\sigma_j \in \{0,1\} \qquad \text{para } j = 1,\ldots,n.$$

Modificación de las funciones de Kernel

La formulación **(6.9)** propone modificar la formulación de las funciones de Kernel de modo de introducir la selección de atributos.

Para las funciones Kernel más importantes (Subsección 5.4) se propone la siguiente modificación:

- Polinomio de grado d: $K_{\sigma}(\vec{x}_i, \vec{x}_s) = \left(\left(\vec{\sigma} * \vec{x}_i \right) \cdot \left(\vec{\sigma} * \vec{x}_s \right) + 1 \right)^d$, $d \in \mathbb{Z}^+$

- Kernel RBF: $K_{\sigma}(\vec{x}_i, \vec{x}_s) = \exp\left(-\frac{\| (\vec{\sigma} * \vec{x}_i) - (\vec{\sigma} * \vec{x}_s) \|^2}{2\rho^2} \right)$, $\rho > 0$

- Función de Transferencia Tangencial:

$$K_{\sigma}(\vec{x}_i, \vec{x}_s) = \tanh\left(\left(\vec{\sigma} * \vec{x}_i \right) \cdot \left(\vec{\sigma} * \vec{x}_s \right) - \theta \right)$$

Algoritmo RFE: Recursive Feature Elimination

La idea es generar un algoritmo backward en el cual, para cada iteración, se elimine el atributo menos relevante en la clasificación:

1. Se define $\vec{\sigma} = (1,...,1)$
2. Se resuelve **(6.9)**
3. se busca el atributo j que aporta menos a la clasificación. Se impone $\sigma_j = 0$
4. se vuelve a 2. hasta un cierto criterio de parada.

Búsqueda del atributo de menor aporte en la clasificación:

En SVM, el valor de

$$W^2(\vec{\alpha}) = \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s K(\vec{x_i}, \vec{x_s}) \qquad \textbf{(6.10)}$$

representa una medida de la habilidad de predicción del modelo (inversamente proporcional al margen) [16]. El algoritmo puede usar esta medida para eliminar los atributos que mantienen esta cantidad pequeña. Para una solución $\vec{\alpha}$ dada, se calcula para cada atributo *p*:

$$W_{-p}^2(\vec{\alpha}) = \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s K(\vec{x_i}^{(-p)}, \vec{x_s}^{(-p)}) \qquad \textbf{(6.11)}$$

donde $\vec{x_i}^{(-p)}$ representa el objeto de entrenamiento *i* sin considerar el atributo *p*. Finalmente se remueve el atributo con menor $| W^2(\vec{\alpha}) - W_{-p}^2(\vec{\alpha}) |$.

Esta medida puede deducirse de la siguiente manera: Se desea resolver el problema Dual **(6.9)** con $\vec{\alpha}$ y $\vec{\sigma}$ como variables de decisión:

$$\max_{\vec{\alpha},\vec{\sigma}} L_D \equiv \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s K_{\vec{\sigma}}(\vec{x_i},\vec{x_s}) \qquad \textbf{(6.12)}$$

s.a

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \qquad \text{para i = 1,...,m.}$$

$$\sigma_j \in \{0,1\} \qquad \text{para j = 1,...,n.}$$

Si se resuelve el problema directamente se obtienen resultados no deseados debido a que se incluyen variables de decisión dentro de la función de Kernel. Para resolver este problema una alternativa es hacerlo por etapas, optimizando sobre $\vec{\alpha}$ fijando $\vec{\sigma}$ y luego optimizando sobre $\vec{\sigma}$ con $\vec{\alpha} = \vec{\alpha}^*$ fijo:

$$\max_{\vec{\sigma}} L_D \equiv \sum_{i=1}^{m} \alpha_i^* - \frac{1}{2}\sum_{i,s=1}^{m} \alpha_i^* \alpha_s^* y_i y_s K_{\vec{\sigma}}(\vec{x_i},\vec{x_s}) \qquad \textbf{(6.13)}$$

s.a

$$\sum_{i=1}^{m} \alpha_i^* y_i = 0$$

$$0 \le \alpha_i^* \le C_1 \qquad \text{para i = 1,...,m.}$$

$$\sigma_j \in \{0,1\} \qquad \text{para j = 1,...,n.}$$

Al considerar $\vec{\alpha}$ fijo tanto el primer término de la función objetivo como las dos primeras restricciones no son necesarias y el problema puede reescribirse de la siguiente manera:

$$\min_{\vec{\sigma}} L_D \equiv \sum_{i,s=1}^{m} \alpha_i^* \alpha_s^* y_i y_s K_{\vec{\sigma}}(\vec{x_i},\vec{x_s}) \qquad \textbf{(6.14)}$$

s.a

$$\sigma_j \in \{0,1\} \qquad \text{para j = 1,...,n.}$$

El problema planteado anteriormente no busca la selección de atributos, por lo que se puede adicionar un término que penalice el uso de atributos:

$$\min_{\vec{\sigma}} L_D \equiv \sum_{i,s=1}^{m} \alpha_i^* \alpha_s^* y_i y_s K_{\vec{\sigma}}(\overrightarrow{x_i}, \overrightarrow{x_s}) + C \cdot \sum_{j=1}^{n} \sigma_j \qquad \textbf{(6.15)}$$

s.a

$$\sigma_j \in \{0,1\} \qquad \text{para } j = 1,\ldots,n.$$

o bien introduciendo una restricción adicional que fije el número deseado de atributos, haciendo innecesario el término penalizador por ser constante. Si se desea eliminar un atributo a la vez de manera iterativa, la formulación queda de la siguiente manera:

$$\min_{\vec{\sigma}} L_D \equiv \sum_{i,s=1}^{m} \alpha_i^* \alpha_s^* y_i y_s K_{\vec{\sigma}}(\overrightarrow{x_i}, \overrightarrow{x_s}) \qquad \textbf{(6.16)}$$

s.a

$$\sum_{j=1}^{n} \sigma_j = n-1$$

$$\sigma_j \in \{0,1\} \quad \text{para } j = 1,\ldots,n.$$

Esta última formulación se resuelve simplemente recorriendo el conjunto de atributo, dejando afuera uno a la vez y evaluando la función objetivo. El menor valor de la función objetivo indica el atributo a eliminar. De esta manera surge la medida de la habilidad de predicción del modelo para el algoritmo RFE-SVM.

Criterio de Parada:

Existen diversos criterios para determinar cuántos atributos deben ser eliminados. En [16] se sugiere elegir los r atributos más relevantes en la clasificación, $r \leq n$, con n el total de atributos. Para ello basta iterar el algoritmo anterior $n - r$ veces.

Sin embargo, el número de atributos adecuados no es una decisión simple que pueda tomarse a priori, por lo que se propone un nuevo criterio de parada que permita identificar el momento en que la pérdida de información por eliminación de atributos es significativa.

La idea es evaluar la clasificación en un conjunto de test usando validación cruzada para evitar un sesgo en las muestras y decidir de acuerdo al menor error asociado al resultado de la clasificación. Partiendo con todos los atributos, es de esperar que, a medida que los atributos que menos aporten en la clasificación sean eliminados, disminuyan los errores de clasificación (de acuerdo a las ventajas mencionadas para la selección de atributos) y por ende la solución mejore. Si muchos atributos son eliminados, y en particular los que aportan significativamente en la clasificación, la solución empeorará, por lo que se puede concluir que existirá una solución que minimiza el número de errores, ya que su pendiente presenta al menos un cambio de signo.

Métodos Tradicionales de Selección de Atributos [18]

Respecto a los métodos tradicionales de selección de atributos, éstos siguen los enfoques Backward o Forward (Sección 4.2.2). Estos métodos son fáciles de usar pero sus resultados dependen de la herramienta utilizada para medir la importancia de cada atributo, además que demandan la realización de entrenamientos sucesivos, lo cual es costoso computacionalmente cuando el número de atributos y objetos es muy grande. Al medir la importancia de cada atributo como la cantidad de información que contiene (entropía) o la cantidad e variabilidad de los datos que puede explicar (Análisis de Componentes Principales) se tiene una medida sesgada de la correlación entre los atributos, puesto que éstos se miden de manera separada.

Por ejemplo, si dos atributos por separado tienen igual importancia en la clasificación y explican un porcentaje γ de la variabilidad de los datos, no se puede asegurar que juntos expliquen 2γ y, en el peor de los casos (si están absolutamente correlacionados), explicarán juntos solamente γ.

6.3 Nuevas metodologías para la Selección de Atributos con SVM no lineal

La formulación propuesta en [5,28] presenta una penalización sobre la función objetivo del problema primal **(P)**, por lo que se encuentra estancada por el hecho de sólo ser capaz de manipular funciones para el problema lineal.

El vector *w,* que es el elemento que permite realizar la selección de atributos en la formulación primal, cumple la siguiente relación de acuerdo a las condiciones de Karush – Kuhn – Tucker:

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x_i} \tag{6.17}$$

Si se realiza la proyección de los elementos a un espacio de Hilbert de mayor dimensión, este vector toma la siguiente forma:

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \phi(\vec{x_i}) \tag{6.18}$$

Si se desea hacer una analogía a la formulación propuesta en [28] y utilizar este vector para penalizar el uso de atributos en la formulación dual, se visualizan las siguientes problemáticas:

- No había sido necesaria especificar la función de proyección hasta ahora, ya que la formulación dependía de exclusivamente de la función de Kernel definida. De acuerdo a esto, si se decide penalizar la norma 1 o la norma 0

(cardinalidad) de *w* se perdería la gran ventaja de la formulación dual, que permite realizar la clasificación en un espacio de mayor dimensión sin explicitar la función de clasificación.

- Sólo es posible definir la función de proyección para Kernels del tipo polinomial. Funciones de proyección asociadas a Kernels inducidos de dimensión infinita, como el Kernel Gaussiano (que por lo general presenta los mejores resultados), no pueden representarse de forma explícita [44].

- Aun si se consigue explicitar la función de proyección, la penalización de atributos se realizaría en el espacio de Hilbert proyectado, donde los atributos no tienen ningún significado. Eliminar atributos en este espacio sólo puede traer ventajas asociadas a una mejor clasificación, pero se pierden las ventajas de la selección de atributos asociadas a la eficiencia computacional y comprensibilidad del modelo.

- Al realizar una eliminación recursiva de los atributos es posible trabajar de mejor manera la correlación existente entre los atributos que mediante la selecciòn de atributos en un problema de optimización, ya que si en cada etapa del algoritmo se evalúa el desempeño de la clasificación, no existe la posibilidad de eliminar dos o más atributos relevantes pero correlacionados entre sí, lo que puede afectar la influencia de estos atributos sobre la variable objetivo.

Debido a estas razones se decide no utilizar el vector *w* como elemento para penalizar el uso de atributos en la formulación no lineal y proponer algoritmos de eliminación recursiva de atributos para clasificación con SVM no lineal. A continuación se proponen estos algoritmos:

6.3.1 Algoritmo de Filtro

Este algoritmo busca ser un método eficaz en identificar y eliminar los atributos que son menos relevantes en describir el comportamiento de la variable dependiente, en este caso el comportamiento del cliente (si fue un buen pagador o no), usando regresiones logísticas sucesivas.

El modelo econométrico regresión logística es usado para estudiar la relación entre una variable dependiente de tipo binaria y una o más variables independientes. La forma genérica de este modelo es [15]:

$$\begin{aligned} y &= f\left(x_1, x_2, ..., x_K\right) + \varepsilon \\ &= x_1\beta_1 + x_2\beta_2 + ... + x_K\beta_K + \varepsilon \end{aligned} \tag{6.19}$$

donde y es la variable dependiente o explicada y $x_1, \ldots, x_K$ son las variables independientes o explicativas.

El término ε es una perturbación aleatoria. Ésta surge por varias razones, principalmente porque no se puede esperar capturar cada influencia sobre una variable, sin importar como se encuentre elaborado el modelo. El efecto neto de estos factores omitidos, sea positivo o negativo, es capturado por esta perturbación.

Este modelo requiere que se cumplan una serie de supuestos:

- Una relación lineal entre la variable dependiente y las variables independientes.

- Que no existe una dependencia lineal entre cualquiera de las variables independientes del modelo. Esto es necesario para la estimación de los parámetros del modelo.

- La esperanza de la perturbación en una observación de la muestra no debe ser función de las variables independientes en cualquier observación, incluido ésta. En otras palabras, las variables independientes no deben contener información valiosa para la predicción de ε.

- Homocedasticidad y ausencia de autocorrelación: Cada perturbación ε_i tiene la misma varianza σ^2 y no se encuentra correlacionada con cualquier otra perturbación ε_j.

- Las perturbaciones siguen una distribución normal de esperanza 0.

Si bien es matemáticamente posible estimar los coeficientes de la ecuación por el procedimiento habitual de mínimos cuadrados, esto conduce a la obtención de resultados no válidos, ya que cuando se calcule la función obtenida para diferentes valores de las variables independientes se obtendrá resultados que, en general, serán diferentes de 0 y 1, los únicos realmente posibles en este caso (variable dependiente binaria), ya que esa restricción no se impone en la regresión lineal, en la que la respuesta puede en principio tomar cualquier valor.

Si se utiliza como variable dependiente la probabilidad p de que la variable dicotómica tome valor 1 (por ejemplo, que el cliente sea buen pagador en el caso de Credit Scoring) y se construye la siguiente función:

$$\ln \frac{p}{1-p} \tag{6.20}$$

Ahora sí se tiene una variable que puede tomar cualquier valor, por lo que se plantea el buscar para ella una ecuación de regresión tradicional:

$$y = \ln \frac{p}{1-p} = x_1\beta_1 + x_2\beta_2 + \ldots + x_K\beta_K + \varepsilon \tag{6.21}$$

Para estimar los parámetros de la regresión se utilizará el método Mínimos Cuadrados (*LS – Least Squares*). Se define el residuo de la observación *i* como la diferencia entre el valor de y_i real y la estimación de y_i, $\hat{y}_i$, dada por $x_i'b$, siendo *b* los estimadores de β. El método minimiza la suma de los residuos al cuadrado de manera irrestricta, con *b* como variable de decisión. Al imponer la condición de primer orden se obtiene:

$$b = (X'X)^{-1} X'Y \qquad \textbf{(6.22)}$$

Notar que el segundo supuesto expuesto asegura la existencia de la matriz inversa de *X'X*.

Se propone realizar el siguiente algoritmo como método de filtro: una regresión logística múltiple con todos los atributos como variables independientes, y la salida *y* como variable dependiente. En cada etapa se eliminará el atributo cuyo coeficiente asociado tenga menor significancia en el modelo (estadísticamente más cercano a cero, de acuerdo al estadígrafo t). La regresión lineal logística permite identificar que variables dependientes (atributos) describen mejor a la variable independiente, cumpliendo la tarea de identificar los atributos relevantes en la clasificación. La eliminación backward de los atributos permite reducir el riesgo de eliminar atributos relevantes pero correlacionados entre sí, lo que puede suceder si se eliminan grupos de atributos.

El criterio de parada del algoritmo se cumple cuando la efectividad del modelo (usando todos los datos como entrenamiento) decrece de manera significativa en una iteración. Este método permite además un análisis estadístico sobre los datos, aportando interpretabilidad al modelo.

Una vez que se tienen los atributos relevantes para el modelo, se efectúa SVM sobre el subconjunto de atributos determinado por el método de filtro, y se identifican los parámetros utilizando validación cruzada.

Para la aplicación de este método el primer supuesto es el más fuerte, ya que en general la relación entre las variables no es lineal o bien se puede lograr una mejor clasificación usando funciones no lineales. Si bien la validez del ajuste de la función estará condicionada por la efectividad del modelo, el método no busca explicar de manera lineal el problema planteado, sino sólo identificar y eliminar las variables que son irrelevantes para explicar la variable dependiente y/o que están muy correlacionadas entre sí.

El siguiente ejemplo gráfico (Figura 6.1) explica lo recién señalado: se consideran 3 atributos (ejes X, Y y Z). Se aplica una regresión logística,

generando una función lineal sobre los objetos (en rojo). Esta función logística permite identificar el atributo en el eje Z como irrelevante, ya que no participa en la función encontrada (tiene un coeficiente asociado estadísticamente cercano a cero) por lo que se elimina este atributo. Los 2 atributos restantes son evidentemente necesarios en la construcción de la función logística por lo que el método de filtro termina. Finalmente se aplica SVM no lineal sobre los atributos relevantes y así obtener una función de clasificación no lineal.

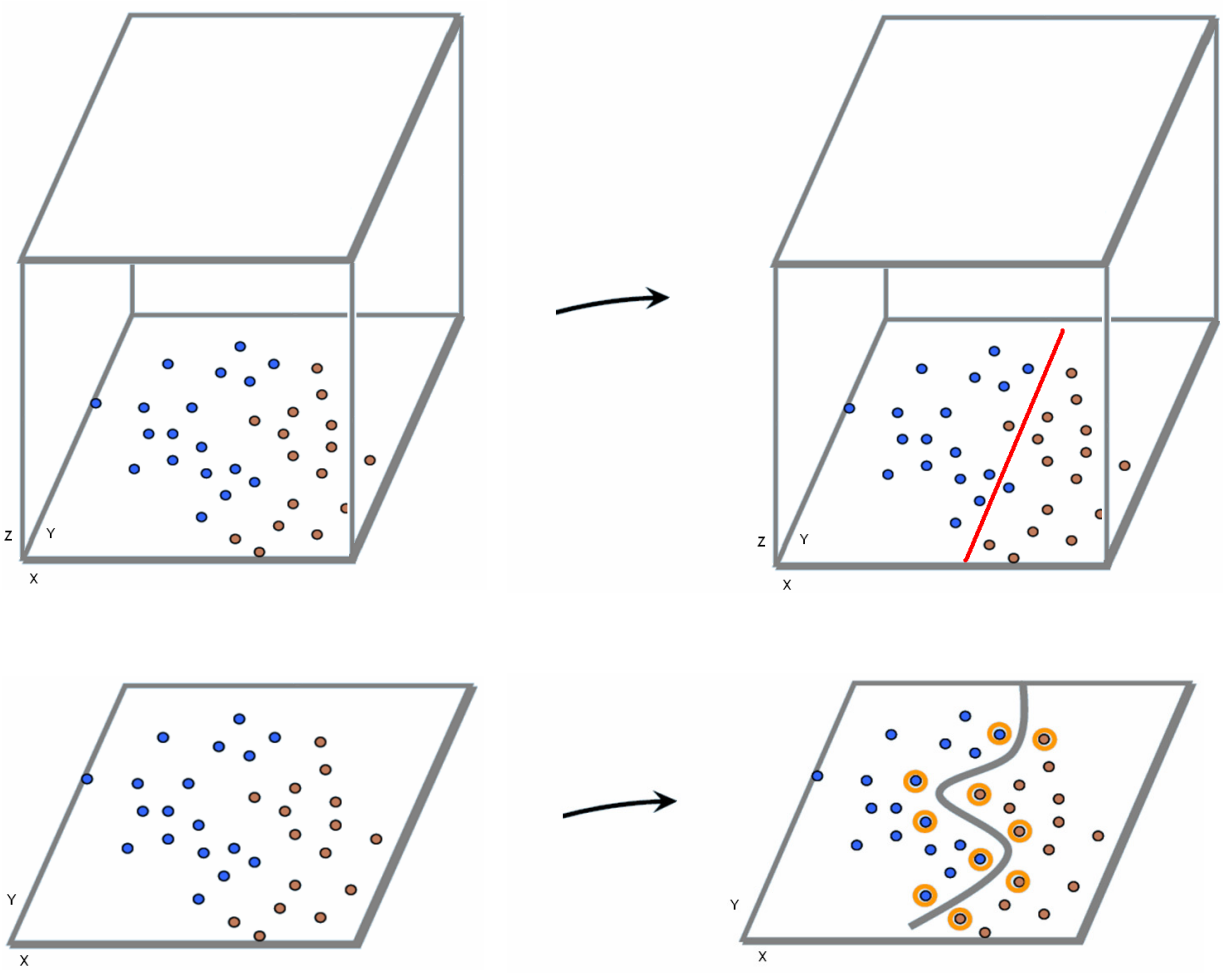

Figura 6.1 Ejemplo Algoritmo de Filtro

6.3.2 Algoritmo Wrapper

Se propone optimizar la formulación anterior considerando todos los atributos (SVM no lineal tradicional), fijar los valores de $\vec{\alpha}$ obtenidos de la optimización y generar una heurística que elimine el atributo que, al ser eliminado, minimiza los costos de clasificación en un conjunto de test. De la misma manera que en los métodos anteriores, se espera que a medida que se eliminen atributos redundantes mejore la predicción del modelo y se llegue a un número óptimo de atributos que minimice los errores de clasificación.

Se considera la formulación Dual con función de Kernel:

$$\max_{\alpha} L_D \equiv \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s K_{\vec{\sigma}}(\vec{x_i}, \vec{x_s}) \qquad \textbf{(6.23)}$$

s.a

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C_1 \qquad \text{para } i = 1,\ldots,m.$$

$$\sigma_j \in \{0,1\} \qquad \text{para } j = 1,\ldots,n.$$

donde $\vec{\sigma}$ es un parámetro.

El algoritmo se describe a continuación:

1. Se realiza la selección del parámetro C del modelo con todos los atributos
2. Se define $\vec{\sigma} = (1,\ldots,1)$.
3. Se realiza una partición aleatoria de los datos, generando un conjunto de entrenamiento y un conjunto de test.
4. Se resuelve **(6.23)** para el conjunto de entrenamiento.

5. Se fija $\vec{\alpha} = \vec{\alpha}^*$ la solución del problema anterior.

6. Se busca el atributo j que aporta menos a la clasificación. Para ello se recorre el conjunto de atributos imponiendo un atributo igual a 0. El atributo eliminado será el que tenga un menor error asociado en el conjunto de test, es decir:

$$E_{(-p)}(\alpha,\sigma) = \sum_{l=1}^{mt} \left| y_l^t - sgn\left(\sum_{i=1}^{m} \alpha_i y_i K_{\bar{\sigma}}(\vec{x}_i^{(-p)}, \vec{x}_l^{t(-p)}) + b \right) \right| \qquad \textbf{(6.24)}$$

   Donde *mt* es el total de elementos en el conjunto de test e $\vec{y}^t$ y $\vec{x}^t$ la información contenida en este conjunto. $\vec{x}_i^{(-p)}$ $(\vec{x}_l^{t(-p)})$ son los objetos del conjunto de entrenamiento (test) cuando se elimina el atributo *p*.

7. Se elimina el atributo *j* ($\sigma_j = 0$) con menor valor de $E_{(-p)}(\vec{\alpha},\vec{\sigma})$.

8. Se vuelve al paso 3. Si en una iteración no se pueden eliminar atributos sin aumentar la cantidad de errores en el test, se termina el algoritmo de selección de atributos.

9. Se seleccionan finalmente los parámetros del modelo.

El hecho de que en cada iteración del algoritmo se elijan aleatoriamente los conjuntos de entrenamiento y test permite reducir el sesgo en la elección de estos conjuntos, evitando el sobreajuste que puede producirse en caso de trabajar con conjuntos fijos. Por otro lado, el orden del algoritmo es equivalente al propuesto en [16], ya que en ambos se lleva a cabo, en cada iteración, la optimización de la formulación no lineal, y se ejecuta posteriormente una heurística que recorre una vez todo el conjunto de atributos. Por esta razón, este algoritmo representa una alternativa al RFE-SVM [16] que puede conseguir mejores resultados.

La Figura 6.2 muestra un esquema del algoritmo anteriormente descrito (en negrita se muestran los vectores):

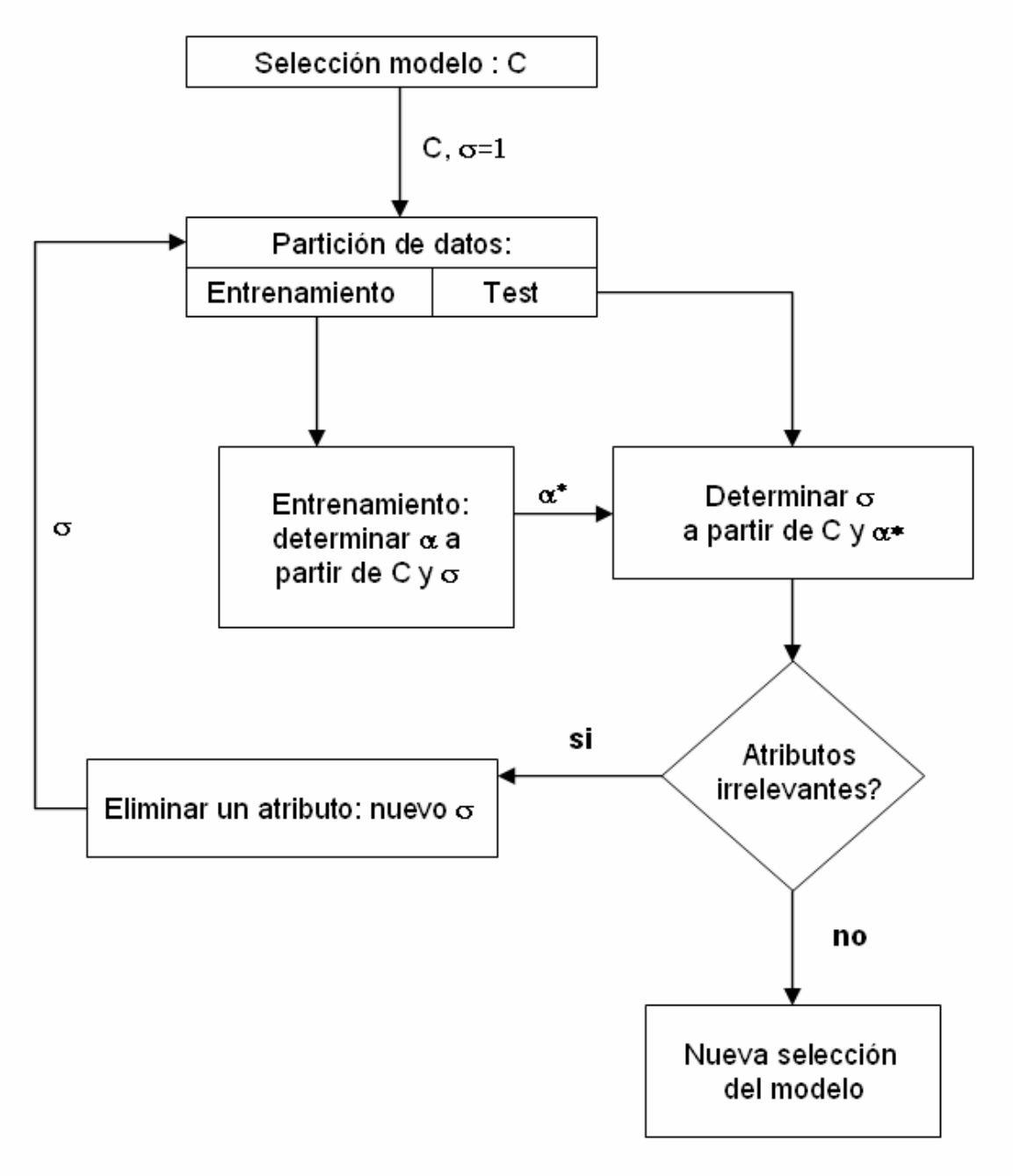

Figura 6.2 Esquema algoritmo Wrapper propuesto

Capítulo 7

Aplicación de metodologías y comparación de resultados

En este capítulo se presentará la aplicación de las metodologías propuestas en una base de datos del repositorio de la Donald Bren School of Information and Computer Sciences de la Universidad de California[4], y a un caso real de Credit Scoring para una entidad financiera nacional. Adicionalmente se describen metodologías conocidas de la literatura, las cuales se aplican al caso con el fin de comparar los métodos propuestos con otros ya probados para estudiar su real efectividad.

7.1 Otros métodos de clasificación utilizados

El análisis de los resultados se completa comparando las metodologías propuestas con otros métodos tradicionales de clasificación: Las Redes Neuronales (algoritmo MLP) y la técnica estadística Regresión Logística. Además, se considerará el método estadístico Análisis de Componentes Principales como un método de filtro para la selección de atributos.

7.1.1. SVM no lineal sin Selección de Atributos

Con el objetivo de medir el desempeño de la selección de atributos, las metodologías propuestas se compararan con la clasifciación obtenida con SVM no lineal e incorporando todos los atributos. Se probarán los Kernels lineal, polinomial y Gaussiano (RBF), utilizando validación cruzada para la selección del modelo.

[4] Fuente: http://www.ics.uci.edu/~mlearn/MLSummary.html, 15 de Mayo de 2007

7.1.2. Redes Neuronales

La Red Neuronal Artificial (ANN) es un método de la inteligencia artificial inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida. Son populares como método de clasificación gracias a su capacidad de modelar funciones de clasificación no lineales. Sin embargo se consideran una "caja negra" y dada la cantidad de parámetros a variar y el riesgo de sobreajuste, se requiere experiencia en el uso de esta herramienta.

7.1.3. Regresión Logística

La Regresión Logística (LR) es usado en la estadística para encontrar una combinación lineal de atributos que separen de mejor manera 2 clases de objetos. Esta herramienta es equivalente a la descrita en la sección 6.3.1, sin embargo, esta vez se usará la función discriminante obtenida sobre un conjunto de entrenamiento para predecir nuevos objetos (pertenecientes a un conjunto de test) utilizando validación cruzada, haciendo el método comparable a los propuestos en esta Tesis.

7.1.4. Método de filtro: ACP

El Análisis de Componentes Principales (ACP) consiste en método de transformación lineal usado para reducir la dimensión de un espacio. ACP involucra el estudio de los valores propios de la matriz varianza-covarianza obtenida de los datos de entrenamiento y la extracción de factores o componentes, que resumen la información de esta matriz y se puede estudiar la relación de éstos con los atributos originales analizando la correlación de ambos conjuntos (componentes y atributos) con la Matriz de Componentes Rotados. Si bien ACP ha demostrado ser un método muy adecuado para la compresión de datos, no es necesariamente una buena herramienta para la selección de atributos, en particular cuando son usados para clasificación supervisada [6]. Aún así, es uno de los métodos de filtro más populares en la literatura.

7.2. Aplicación: Wisconsin Breast Cancer Database

Wisconsin Breast Cancer es una base de datos que pertenece al UCI Repository de la Donald Bren School of Information and Computer Sciences de la Universidad de California. El uso de las bases de datos de este repositorio es usual en la investigación de nuevos métodos de aprendizaje supervisado y permite la comparación de resultados entre las diferentes técnicas. Se eligió esta base de datos en particular debido a que el número de objetos y atributos de esta base resulta adecuado para probar SVM y selección de atributos.

7.2.1. Descripción de la base de datos

Wisconsin Breast Cancer fue creada por el Dr. William H. Wolberg del General Surgery Department de la Universidad de Wisconsin y por W. Nick Street y Olvi L. Mangasarian del Computer Sciences Department de la misma Universidad, y donada en Noviembre de 1995.

La base de datos cuenta con 569 instancias (212 tumores malignos y 357 benignos) y 32 atributos: un elemento identificador (el cual no se usa para la clasificación), el diagnóstico del tumor (1 si es benigno, -1 si es maligno) y 30 atributos continuos con la descripción del tumor:

Los atributos son calculados a partir de una imagen digital del aspirado con aguja fina (FNA) de una masa mamaria. Ellos describen características de los núcleos celulares presentes en la imagen.

Diez rasgos en valor real son computados para cada núcleo celular:

1. Radio (promedio de distancias del centro hasta puntos en el perímetro)
2. Textura (desviación estándar de valores en escala de grises)
3. Perímetro del núcleo
4. Área del núcleo
5. Uniformidad (variación local de longitudes radiales)
6. Compacidad (perímetro^2 / área - 1)
7. Concavidad (profundidad de porciones cóncavas del contorno)

8. Puntos cóncavos (número de porciones cóncavas del contorno)
9. Simetría
10. Dimensión fractal

La media, el error estándar, y el "peor" o mayor (promedio de los tres valores superiores) de estos rasgos fueron computados para cada imagen, resultando en 30 rasgos. Por ejemplo, el campo 3 es media radial, el campo 13 es error estándar del radio, el campo 23 es el "peor" radio.

Cada valor de un atributo se escala entre 0 y 1 y se re-codifica a cuatro cifras significativas. Para escalar la base en este rango se usa la siguiente fórmula:

$$\widetilde{X}_{ij} = \frac{X_{ij} - MIN_j(X)}{MAX_j(X) - MIN_j(X)} \quad \textbf{(7.1)}$$

Donde X_{ij} es el valor original y $MAX_j(X)$ y $MIN_j(X)$ son el máximo y mínimo valor que toma el atributo *j* respectivamente.

La base de datos no cuenta con valores perdidos.

7.2.2. Resultados de los modelos

7.2.2.1. SVM no lineal

Se aplica a la base de datos SVM no lineal con todos los atributos, usando Kernel Lineal, Kernel Polinomial y Kernel Gaussiano (RBF) y buscando la mejor configuración de los parámetros con validación cruzada con 10 particiones.

Para el Kernel Lineal se requiere sólo un parámetro, C, que representa la cota superior del vector α. Se varía este parámetro usando 21 valores seleccionados del trabajo [29]:

C={0.1;0.3;0.5;0.7;1;10;20;30;40;50;60;70;80;90;100;200;300;400;500;700;1000}

El mejor resultado se obtiene para un coeficiente C=100, con un promedio de un 94,55% de efectividad en la validación cruzada. En la Tabla 10.1 del Anexo 10.1.1 se muestra el detalle de los resultados obtenidos.

El Kernel Polinomial de grado *d* está caracterizada por la siguiente ecuación:

$$K(\vec{x}_i, \vec{x}_s) = \left(\vec{x}_i \cdot \vec{x}_s + 1\right)^d \ , \ d \in \mathbb{Z}^+ \qquad \textbf{(7.2)}$$

Se decide variar el parámetro *d* de acuerdo a lo sugerido en [27], es decir, entre los valores {2,3,4,5,6,7,8,9}. La mejor configuración {*d,C*} se tiene para *d*=3 C=80, con un con un promedio de un 96,49% de efectividad en la validación cruzada. En la Tabla 10.2 del Anexo 10.1.1 se muestran los resultados obtenidos en la validación cruzada.

El último Kernel utilizado para realizar la clasificación considerando todos los atributos es el Kernel Gaussiano o RBF, el cual viene dado por la siguiente expresión:

$$K(\vec{x}_i, \vec{x}_s) = \exp\left(-\frac{\| \vec{x}_i - \vec{x}_s \|^2}{2\rho^2}\right) \ , \rho > 0 \qquad \textbf{(7.3)}$$

El parámetro a variar es ρ y de acuerdo a lo realizado en [27], es decir, se utilizan los valores {0.1,0.5,1,2,3,4,10,20,100}. La mejor configuración {ρ,*C*} se tiene para ρ=2 C=100, con un con un promedio de un 98,25% de efectividad en la validación cruzada. En la Tabla 10.3 del Anexo 10.1.1 muestra el detalle de los resultados obtenidos en la validación cruzada.

Todos estos experimentos fueron realizados con el software Matlab versión 7.2 y la biblioteca Spider para Matlab[5].

[5] Fuente: http://www.kyb.tuebingen.mpg.de/bs/people/spider/, 15 de Julio de 2007

7.2.2.2. Métodos de Filtro

Dentro de los métodos de filtro se consideran el ACP y el algoritmo propuesto, LR-SVM.

Filtro ACP

El primer paso del método ACP consiste en extraer los factores a partir de la matriz de correlaciones. Para esta base de datos se extraen 6 factores o componentes, los cuales explican un 88,579% de la varianza total. Como criterio de extracción de factores se decide considerar los componentes cuyos valores propios asociados sean mayores a 1.

Con el fin de identificar los atributos a eliminar, se asocian éstos a cada componente extraído usando la Matriz de Componentes Rotados (rotación de ejes Varimax) y se elijen los atributos que presenten una mayor correlación (en módulo) con los componentes. Dado que no existe un criterio claro sobre la cantidad de atributos a considerar, se decide considerar los 20 atributos más relevantes, es decir, con una mayor correlación con los componentes.

Para finalizar se efectúa SVM no lineal sobre el subconjunto de atributos seleccionado. Considerando que el Kernel RBF presenta un mejor desempeño que el Kernel Polinomial y el Kernel Lineal, se considerará éste para la tarea de clasificación. La variación de parámetros para encontrar la mejor configuración $\{\rho, C\}$ se efectúa a igual que considerando todos los atributos.

El mejor resultado se obtiene para la configuración $\rho = 2$, C=80, con un porcentaje de eficiencia del Test de un 97,72% en la validación cruzada. (Tabla 10.4 del Anexo 10.1.1)

El Análisis de Componentes Principales se efectuó con el software SSPS versión 13.0, mientras que la obtención de la configuración óptima de parámetros se realizó el software Matlab y la biblioteca Spider.

Filtro con Regresión Lineal

La primera parte del método consiste efectuar una regresión logística con todos los atributos, se elimina el atributo menos relevante, se vuelve a realizar una regresión logística (sin considerar el atributo eliminado) y se elimina nuevamente un atributo. Este proceso se repite hasta notar una disminución en los coeficientes de bondad de ajuste del modelo.

Se eliminaron 4 atributos con este método de filtro. En la última iteración del método todos los atributos eran estadísticamente significativos.

El método termina con la búsqueda de la configuración óptima con SVM no lineal y Kernel RBF sobre el subconjunto de atributos seleccionado. El mejor resultado se obtiene para la configuración $\rho=2$, $C=40$, con un porcentaje de eficiencia del Test de un 98,07% en la validación cruzada (Tabla 10.5 del Anexo 10.1.1).

Cabe destacar la ventaja de este método por sobre el señalado anteriormente en el sentido de entregar un criterio de parada, sin embargo la cantidad de atributos irrelevantes identificados es baja.

Las regresiones logísticas se efectuaron con el software Weka versión 3.4, mientras que la obtención de la configuración óptima de parámetros se realizó con el software Matlab y la biblioteca Spider.

7.2.2.3. Métodos Wrapper

Los métodos Wrapper que se consideran son la modificación del algoritmo RFE-SVM y la propuesta descrita en la sección 6.3.2.

Algoritmo RFE-SVM

La primera etapa del método consiste en establecer un ranking de los atributos, y establecer la secuencia con la cual los atributos deben ser eliminados. Usando el criterio de parada descrito en la sección 6.2 se eliminaron 11 atributos.

La mejor configuración que se consigue con los parámetros $\rho=3$, $C=100$, con un porcentaje de eficiencia del Test de un 97,89% en la validación cruzada (Tabla 10.6 del Anexo 10.1.1). Para establecer el ranking de los atributos y determinar la mejor configuración de parámetros su utilizó Matlab y la biblioteca Spider.

Algoritmo Wrapper de desarrollo personal

El método consiste también en una eliminación recursiva de atributos, donde en cada iteración se particiona la base (un 70% de los elementos en un conjunto de entrenamiento y un 30% de ellos en un conjunto de test aproximadamente), se efectúa SVM no lineal sobre el conjunto de entrenamiento y se evalúa la solución en el conjunto de test, recorriendo el conjunto de atributos e imponiendo que un atributo a la vez no sea considerado. Se elimina el atributo para el cual, al ser eliminado, se obtenga una menor cantidad de errores en el test y el algoritmo para cuando no se encuentren mejoras en una iteración.

Se seleccionaron 22 atributos, y la mejor configuración encontrada se consigue con los parámetros $\rho=4$, $C=100$. El porcentaje de eficiencia del test es en promedio de un 98,25% en la validación cruzada, como se observa en la Tabla 10.7 del en el Anexo 10.1.1. El algoritmo fue programado en lenguaje para Matlab y usando la biblioteca Spider.

7.2.2.4. Otros métodos de clasificación

A continuación se muestran los resultados obtenidos por los métodos descritos en la sección 7.1.2 para la tarea de clasificación binaria: el método estadístico Regresión Logística y Redes Neuronales.

La Regresión Logística es un método simple para el cual no se requiere una especial calibración de parámetros. El resultado de la clasificación es un acierto del modelo de un 93,67% usando validación cruzada, como se muestra en el Anexo 10.3.1. Este método fue efectuado con el software Weka versión 3.4.

Para las Redes Neuronales se requiere un trabajo importante en la calibración de los parámetros, el cual debe hacerse con validación cruzada. La mejor configuración se consigue con el algoritmo de aprendizaje MLP (Multilayer Perceptron), 32 neuronas en la capa oculta, una tasa de aprendizaje de 0.3, momento de 0.6 y sin considerar decaimiento de pesos. El porcentaje de eficiencia logrado es de un 96,66% con validación cruzada. El software usado para desarrollar este método fue Weka versión 3.4. El Anexo 10.3.1. muestra un mayor detalle de los resultados.

7.3. Aplicación: Asignación de Créditos

Para aplicar las metodologías propuestas se utiliza la base de datos de una entidad crediticia nacional.

7.3.1. Descripción de la base de datos

La base de datos final cuenta con 1464 instancias (767 clientes buenos y 697 clientes malos) y 49 atributos (29 variables continuas y 20 variables binarias).

La información de los clientes considerada para la clasificación fue recogida entre los años 2000 y 2005. Por razones de privacidad no se realiza una descripción de los atributos que se consideran para la clasificación.

La primera etapa del preprocesamiento corresponde al análisis de calidad de datos. Se analizan dos dimensiones de la calidad de datos: datos faltantes (NULL) y análisis de concentración. En el análisis de datos faltantes se detectan aquellas variables que por contar con una alta proporción de datos faltantes (no existe información), no son aptas para formar parte de un modelo predictivo. Análogamente, en el análisis de concentración se detectan aquellas variables que por presentar un alto nivel de concentración en un valor específico, se excluyen de los análisis estadísticos futuros y de los modelos que se construirán (e.g. la variable nacionalidad está concentrada en un 100% en el valor ‘chilena’, por lo no tiene sentido alguno incorporarla en un modelo predictivo). Como resultante de esta etapa se tiene un conjunto de variables cuya calidad de datos

permite continuar a la etapa siguiente, eliminándose de los análisis las demás variables.

La siguiente etapa es el análisis estadístico univariado del grado de relación de cada una de las variables explicativas que sobreviven el análisis de calidad de datos, con la variable objetivo (dependiente) del problema (MALO/BUENO). Ejemplo: ¿Está relacionada la variable edad con el hecho que un crédito resulte MALO? ¿Influye el género de una persona en el hecho que un crédito resulte MALO? Se emplean pruebas estadísticas de uso estándar para este tipo de problemas para responder preguntas como las planteadas: para variables numéricas, se utiliza el test K-S (Kolmogorov-Smirnov) para evaluar si la diferencia máxima en la distribución acumulada de una determinada variable (e.g. edad), es significativamente distinta para créditos BUENOS vs. MALOS; análogamente, para variables nominales (e.g. estado civil) se utiliza el test chi-cuadrado de Pearson para evaluar si cada variable en particular es o no independiente de la variable objetivo. Como resultante de esta etapa se seleccionan las variables con mayor grado de asociación univariada con la variable objetivo, las que se transforman en variables candidatas a ser incorporadas en modelos multivariados en la siguiente etapa.

La base de datos fue limpiada de valores perdidos mediante la eliminación de las filas que contenían estos valores.

La base fue balanceada para obtener un número semejante de observaciones de clientes buenos y malos, ya que el número original de instancias de clientes buenos es mucho mayor al de clientes malos. Este balance se efectúa realizando un muestreo aleatorio para ambas clases, seleccionando en cada una aproximadamente 700 observaciones.

Los atributos nominales fueron transformados a variables dicotómicas. Cada valor de un atributo continuo fue escalado entre 0 y 1 y re-codificado a cuatro cifras significativas. Para escalar la base en este rango se usa la siguiente fórmula:

$$\widetilde{X}_{ij} = \frac{X_{ij} - MIN_j(X)}{MAX_j(X) - MIN_j(X)} \quad \textbf{(7.4)}$$

donde X_{ij} es el valor original y $MAX_j(X)$ y $MIN_j(X)$ son el máximo y mínimo valor que toma el atributo *j* respectivamente.

7.3.2. Resultados de los modelos

7.3.2.1. SVM no lineal

Para esta base de datos se aplicó SVM en su versión no lineal y considerando todos los atributos, usando los Kernel Lineal, Kernel Polinomial y Kernel Gaussiano (RBF) y buscando la mejor configuración de los parámetros con validación cruzada con 10 particiones (10-fold crossvalidation).

Para el Kernel Lineal se requiere un solo parámetro, C, que representa la cota superior del vector α. Se varía este parámetro de la misma manera que en el caso Wisconsin Breast Cancer.

El mejor resultado se obtiene para un coeficiente C=30, con un promedio de un 71,10% de efectividad en la validación cruzada. En la Tabla 10.8 del Anexo 10.1.2 se muestra el detalle de los resultados obtenidos. La Figura 7.1 muestra la evolución del porcentaje de acierto del Test en la validación cruzada al variar C.

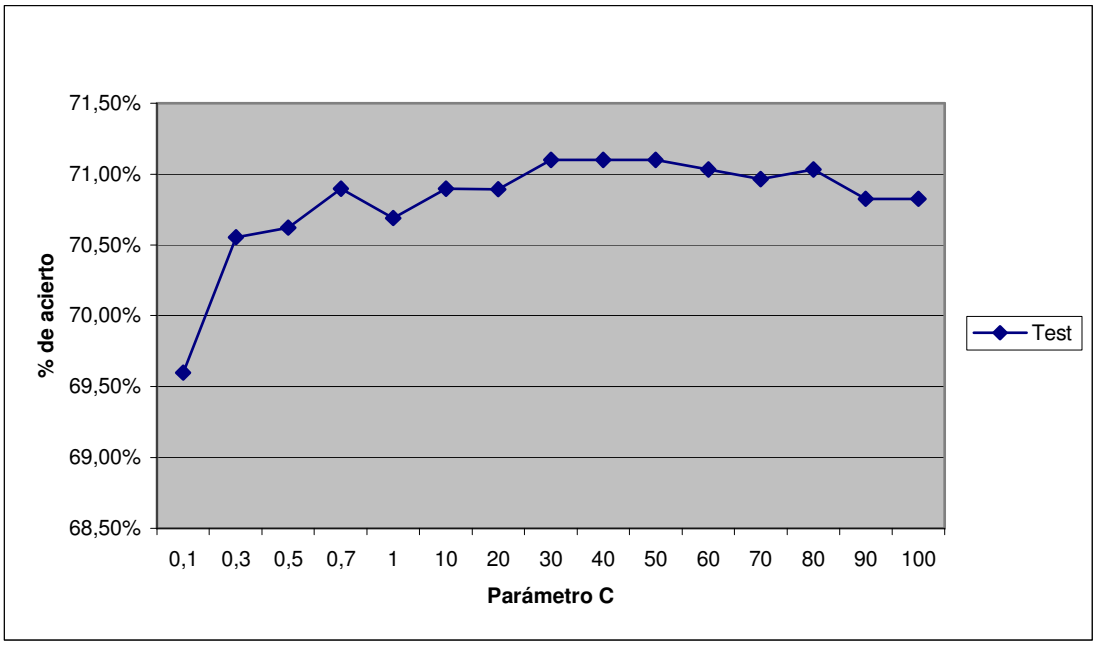

Figura 7.1: Evolución acierto Test respecto a C, SVM lineal

El Kernel Polinomial de grado *d* está caracterizada por la siguiente ecuación:

$$K(\vec{x}_i, \vec{x}_s) = \left(\vec{x}_i \cdot \vec{x}_s + 1\right)^d \text{ , } d \in \mathbb{Z}^+ \qquad (7.5)$$

Se varia el parámetro *d* de acuerdo a lo sugerido en [27], es decir, entre los valores {2,3,4,5,6,7,8,9}. La mejor configuración {*d,C*} se tiene para *d*=2 C=0,05; con un con un promedio de un 75,27% de efectividad en la validación cruzada. En la Tabla 10.9 del Anexo 10.1.2 se muestra el detalle de los resultados obtenidos en la validación cruzada. Las Figuras 7.2, 7.3, 7.4, 7.5 muestran la evolución del porcentaje de acierto del Test en la validación cruzada al variar C para d=2, 3, 4 y 5 respectivamente.

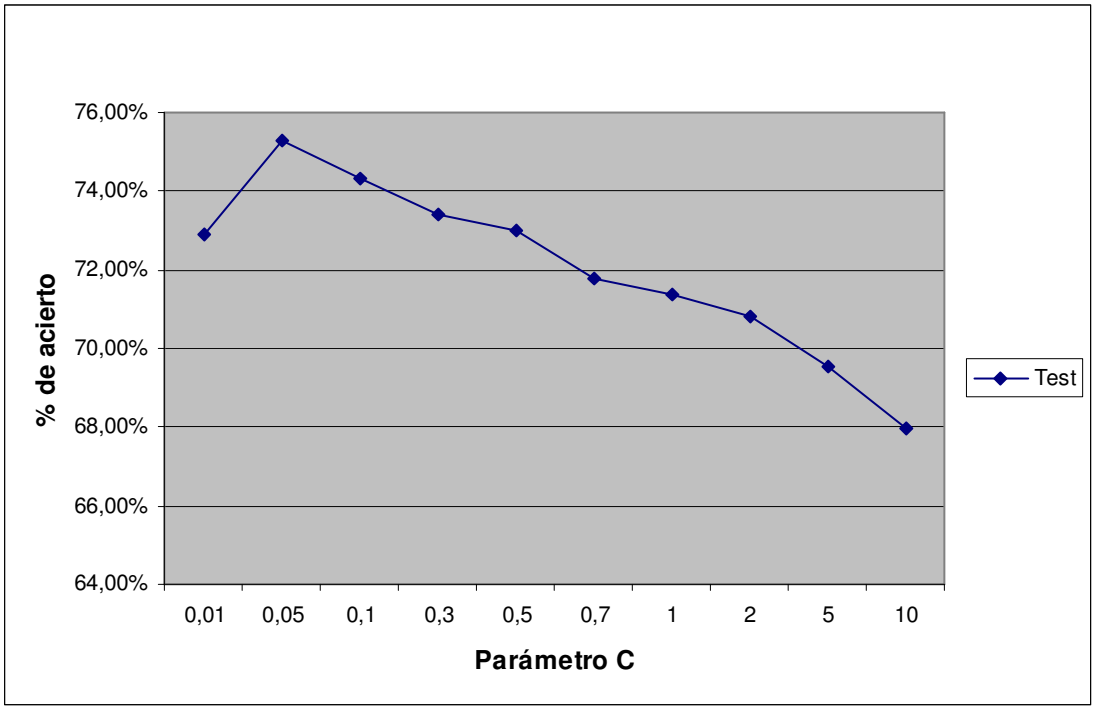

Figura 7.2: Evolución acierto Test respecto a C, SVM Polinomial, d=2

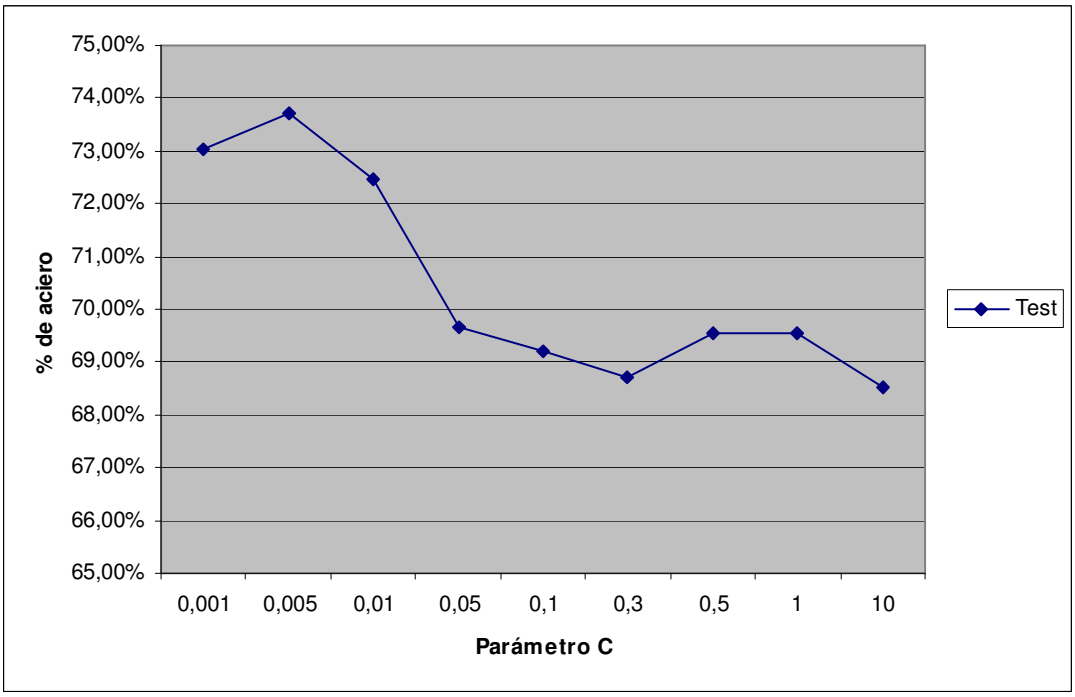

Figura 7.3: Evolución acierto Test respecto a C, SVM Polinomial, d=3

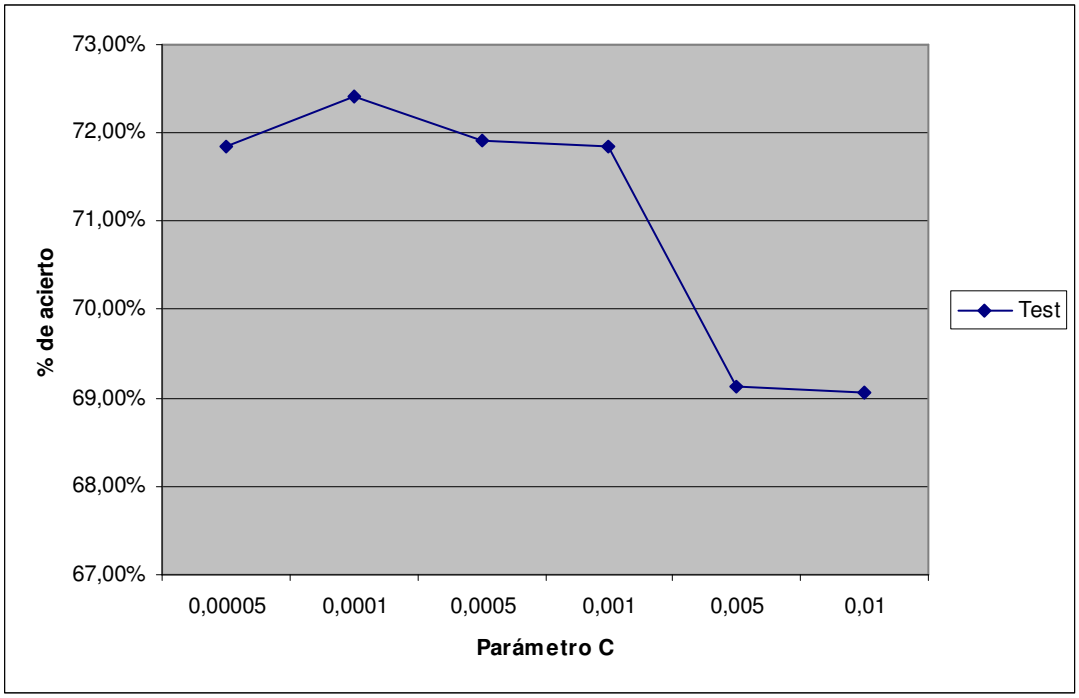

Figura 7.4: Evolución acierto Test respecto a C, SVM Polinomial, d=4

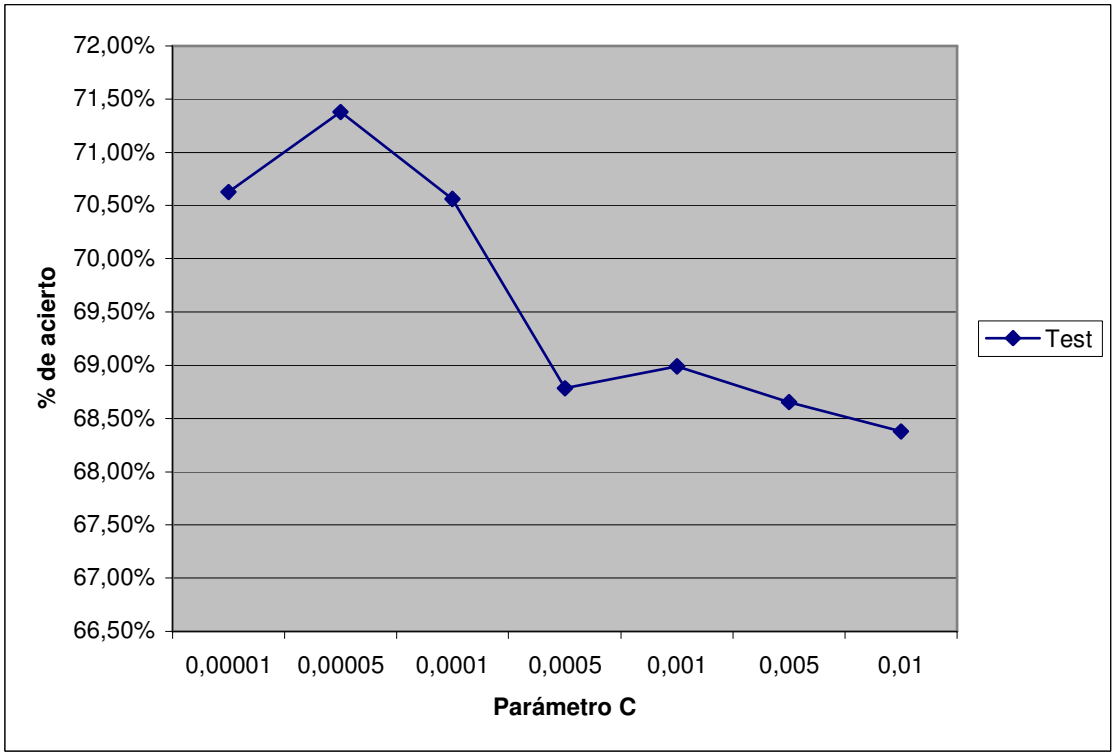

Figura 7.5: Evolución acierto Test respecto a C, SVM Polinomial, d=5

El último Kernel considerado en esta subsección es el Kernel Gaussiano o RBF, el cual viene dado por la siguiente expresión:

$$K(\vec{x}_i, \vec{x}_s) = \exp\left(-\frac{\| \vec{x}_i - \vec{x}_s \|^2}{2\rho^2}\right), \rho > 0 \qquad (7.6)$$

El parámetro a variar es ρ y de acuerdo a lo realizado en [27], es decir, se utilizan los valores {0.1,0.5,1,2,3,4,10}. La mejor configuración $\{\rho, C\}$ se tiene para ρ=4 C=30, con un con un promedio de un 75,54% de efectividad en la validación cruzada. En la Tabla 10.10 del Anexo 10.1.2 muestra el detalle de los resultados obtenidos en la validación cruzada. Las Figuras 7.6 y 7.7 muestran la evolución del porcentaje de acierto del Test en la validación cruzada para la mejor configuración de C al variar ρ, y la evolución del acierto del Test al variar C para ρ=4 respectivamente.

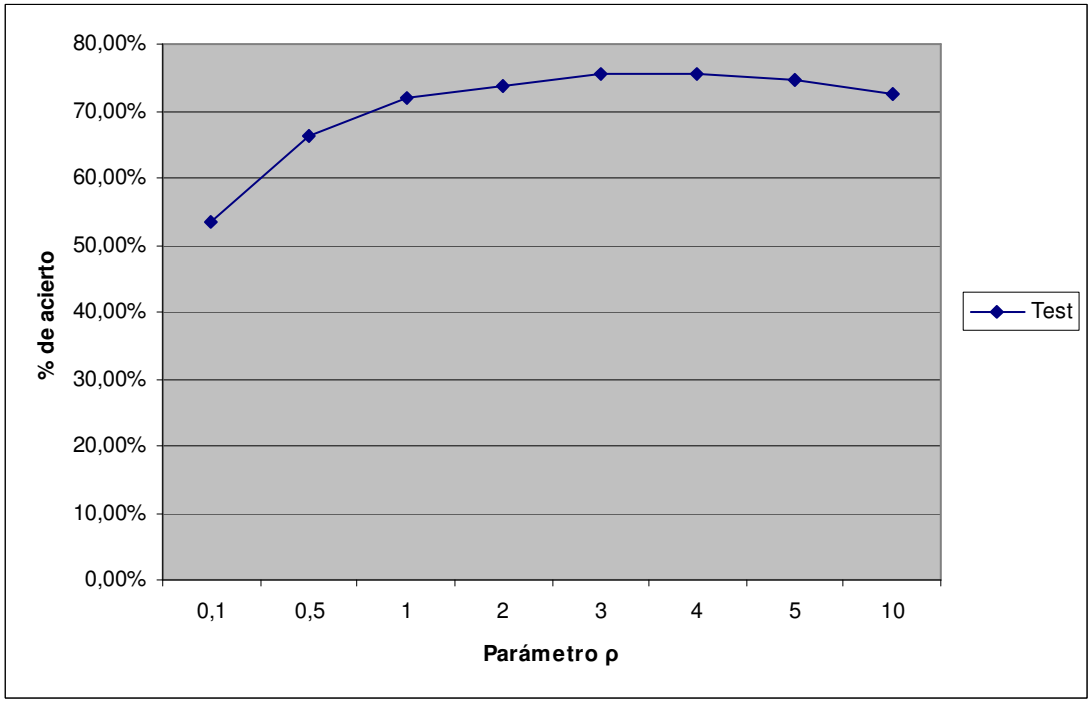

Figura 7.6: Evolución acierto Test respecto a ρ para mejor C, SVM RBF

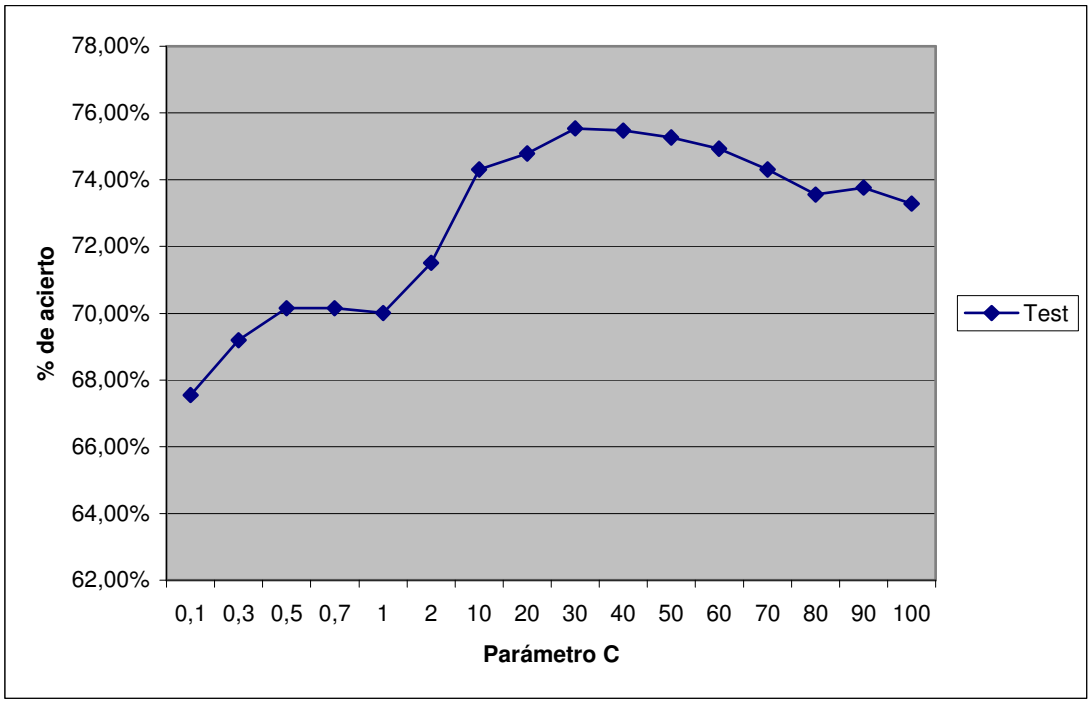

Figura 7.7: Evolución acierto Test respecto a C para $\rho = 4$, SVM RBF

Todos estos experimentos fueron realizados con el software Matlab y la biblioteca Spider para Matlab.

7.3.2.2. Métodos de Filtro

Dentro de los métodos de filtro se consideran el ACP y el algoritmo propuesto, LR-SVM.

Filtro ACP

El primer paso del método ACP consiste en extraer los factores a partir de la matriz de correlaciones. Para esta base de datos se extraen 13 factores o componentes, los cuales explican un 64,449% de la varianza total. Como criterio de extracción de factores se decide considerar los componentes cuyos valores propios asociados sean mayores a 1.
La eliminación de atributos se realiza asociando éstos a cada componente extraído usando la Matriz de Componentes Rotados (rotación de ejes Varimax) y

se eligiendo los atributos que presenten una mayor correlación (en módulo) con los componentes. Se decide considerar los 30 atributos más relevantes, es decir, con una mayor correlación con los componentes.

Para finalizar se efectúa SVM no lineal sobre el subconjunto de atributos seleccionado. Se considera nuevamente el Kernel RBF por presentar mejores resultados que los otros Kernels. El mejor resultado se obtiene para la configuración ρ =4, C=70, con un porcentaje de eficiencia del Test de un 71,92% en la validación cruzada (Anexo 10.1.2, Tabla 10.11).

El Análisis de Componentes Principales se efectuó con el software SSPS versión 13.0, mientras que la obtención de la configuración óptima de parámetros se realizó el software Matlab y la biblioteca Spider.

Filtro con Regresión Logística

La primera etapa del método consiste en eliminar de manera iterativa (backward) los atributos irrelevantes de acuerdo a los coeficientes asociados a cada atributo.

El método termina con la búsqueda de la configuración óptima con SVM no lineal y Kernel RBF sobre el subconjunto de atributos seleccionado. El mejor resultado se obtiene para la configuración ρ =5, C=30, con un porcentaje de eficiencia del Test de un 74,31% en la validación cruzada (Anexo 10.1.2, Tabla 10.12)

Cabe destacar la ventaja de este método por sobre el señalado anteriormente en el sentido de entregar un criterio de parada que permite determinar a partir de que punto el algoritmo empieza a perder información relevante a causa de la eliminación de atributos.

Las regresiones lineales se efectuaron con el software Weka versión 3.4, mientras que la obtención de la configuración óptima de parámetros se realizó el software Matlab y la biblioteca Spider.

7.3.2.3. Métodos Wrapper

Los métodos Wrapper que se consideran son la modificación del algoritmo RFE-SVM y la propuesta descrita en la sección 6.3.2.

Algoritmo RFE-SVM

La primera etapa del método consiste en establecer un ranking de los atributos, y establecer la secuencia con la cual los atributos deben ser eliminados.

Usando el criterio de parada descrito en el punto 6.2 se seleccionan 25 atributos. La mejor configuración que se consigue con los parámetros $\rho = 3$, C=40, con un porcentaje de eficiencia del Test de un 73,97% en la validación cruzada (Anexo 10.1.2, Tabla 10.13). Para establecer el ranking de los atributos y determinar la mejor configuración de parámetros su utilizó Matlab y la biblioteca Spider.

Algoritmo Wrapper de desarrollo personal

Para este método se seleccionaron 24 atributos, y la mejor configuración encontrada se consigue con los parámetros $\rho = 2$, C=60. El porcentaje de eficiencia del test es en promedio de un 75,48% en la validación cruzada (Anexo 10.1.2, Tabla 10.14). El algoritmo fue programado en lenguaje para Matlab y usando la biblioteca Spider.

7.3.2.4. Otros métodos de clasificación

Al aplicar Regresión Logística sobre la base de datos de Credit Scoring se consigue una efectividad de un 71,11% usando validación cruzada, como se muestra en el Anexo 10.2.2.

Para el caso de las Redes Neuronales, la mejor configuración se consigue con el algoritmo de aprendizaje MLP (Multilayer Perceptron), 51 neuronas en la capa oculta (número de atributos más número de clases), una tasa de aprendizaje de 0.3 y momento 0.2, sin considerar decaimiento de pesos. El porcentaje de eficiencia logrado con validación cruzada es de un 70,15%.

Capítulo 8

Extracción de Reglas para Credit Scoring

En este proyecto de Tesis, y al igual que en muchos otros problemas de clasificación, SVM no lineal ha sido usado exitosamente en un amplio rango de aplicaciones. El buen desempeño de este método se debe a la transformación implícita del problema original a un espacio de mayor dimensión, lo que se traduce en un clasificador no lineal en el espacio de origen. Este clasificador es, sin embargo, una compleja función matemática, incomprensible para las personas.

La aplicación Credit Scoring requiere tanto de una clasificación precisa como de comprensibilidad. Para superar esta limitancia de SVM, es posible extraer reglas de la base entrenada que sean interpretables y que le permitan a la entidad crediticia tomar decisiones táctico-estratégicas. Las técnicas de extracción de reglas buscar “abrir la caja negra” que representa la clasificación con SVM y generar una descripción interpretable con aproximadamente el mismo poder predictivo que el modelo mismo.

En este capítulo se presenta un método que cumpla este objetivo y se aplica a un caso de Credit Scoring para extraer reglas a partir de la clasificación con mejor desempeño obtenida del capítulo anterior, incorporando la selección de atributos.

8.1. Técnicas para la extracción de reglas

En [26] se señalan dos enfoques posibles para la extracción de reglas a partir de métodos de clasificación: el enfoque de descomposición, el cual se relaciona directamente como el proceso interno de SVM y la construcción del hiperplano; y el enfoque pedagógico, que considera el entrenamiento del algoritmo como una "caja negra", extrayendo las reglas directamente de las entradas y salidas del algoritmo. El supuesto detrás de estas técnicas es que el modelo entrenado puede representar mejor los datos que el conjunto inicial.

El poder de las reglas extraídas depende del lenguaje usado para expresarlas. Los tipos de reglas más relevantes son del tipo proposicional (expresiones del tipo si… entonces…), reglas M-de-N (si se cumplen al menos M de las N condiciones entonces…) y reglas difusas que permiten mayor flexibilidad.

En este proyecto se trabajará con reglas del tipo proposicional, usando el algoritmo árbol de decisión C4.5.

8.2. Algoritmo C4.5

El algoritmo C4.5 es un tipo de árbol de decisión basado en el concepto de entropía [32]. Sea p_1 (p_0) la proporción de ejemplos de la clase 1 o buenos pagadores (-1 o malos pagadores) en una muestra *S*. La *entropía* de *S* se calcula de la siguiente manera:

$$\textit{Entropía(S)} = -p_1 \log_2(p_1) - p_0 \log_2(p_0), \qquad \textbf{(8.1)}$$

donde p_0=1- p_1. La entropía se usa como medida de cuan bueno es un atributo para dividir el conjunto de datos. Básicamente la entropía es una medida del desorden en los datos con respecto a las clases, ya que es igual a 1 cuando p_0=p_1=0,5 (máximo desorden) y 0 cuando p_0 o bien p_1 son cercanos a 0 (mínimo desorden, todas los objetos pertenecen a la misma clase). La *Ganancia(S,*x_j*)* se

define como la reducción esperada de la entropía al dividir la muestra *S* usando el atributo x_j:

$$\text{Ganancia}(S, x_j) = \text{Entropía}(S) - \sum_{v \in valores(x_j)} \frac{card(S_v)}{card(S)} \cdot \text{Entropía}(S_v) \quad \textbf{(8.2)}$$

donde $valores(x_j)$ representa el conjunto de todos los posibles valores del atributo x_j, S_v el subconjunto de *S* donde x_j toma valor v y $card(S_v)$ el número de objetos en S_v. El algoritmo C4.5 aplica una normalización y usa un *Índice de Ganancia (IG)* como criterio para elegir que atributo se ocupará para dividir el subconjunto S en un nodo dado. El IG se define de la siguiente manera:

$$\text{IG}(S, x_j) = \frac{\text{Ganancia}(S, x_j)}{-\sum_{k \in valores(x_j)} \frac{card(S_k)}{card(S)} \log_2 \frac{card(S_k)}{card(S)}} \quad \textbf{(8.3)}$$

Este algoritmo inductivo se aplica sobre los datos donde la salida se modifica por los valores predichos por la mejor solución de SVM, incluyendo la selección de atributos, y de esta manera el árbol de decisión aproxima al método SVM. Un problema que puede surgir es que a medida que el árbol se expande. Menos objetos habrán disponibles en los nodos para dividir. Esto se soluciona con un adecuado criterio de poda sobre el árbol para evitar sobreajuste. Cabe mencionar también que la base de datos de Credit Scoring es extensa en objetos en relación al número de atributos (1464 observaciones y 49 atributos, 24 atributos la mejor política de selección de atributos) por lo que se garantiza una cantidad significativa de objetos en cada nodo.

8.3. Resultados

Se aplicó el algoritmo descrito sobre la base de datos de Credit Scoring con la mejor selección de atributos obtenida, que corresponde al algoritmo Wrapper de desarrollo personal con 24 atributos elegidos y la configuración $\rho = 2$, C=60 (Kernel RBF). El método fue ejecutado con el software Weka 3.4.

El modelo se entrenó y testeó sobre todo el conjunto de observaciones (1464 en total) y se consiguió una precisión de un 85,24%. Los parámetros usados en la selección de modelo son un factor de confianza (“confidence factor”, parámetro de poda) de 0.01 y un número mínimo de 10 observaciones en una hoja. El tamaño del árbol es de 33, con 17 hojas.

En la Figura 8.1 se muestra un esquema del árbol de decisión C4.5 descrito por sus nodos, hojas y los atributos considerados para la división. En verde se muestran las hojas cuyo elemento predominante son los clasificados como buenos pagadores (“a”) y en rojo las hojas cuyo elemento predominante son los clasificados como malos pagadores (“b”). En el Anexo 10.4 se muestra el árbol descrito como un conjunto de reglas (por razones de privacidad se reemplaza el nombre del atributo por un número).

Se concluye que este método puede describir de manera bastante precisa (85% de efectividad) la salida del método SVM, permitiendo llevarla a reglas sencillas como “los clientes que tienen 24 (a24>0, primer nodo, ver Anexo 10.4) serán probablemente malos pagadores”, lo que es un patrón muy útil al momento de tomar decisiones estratégicas.

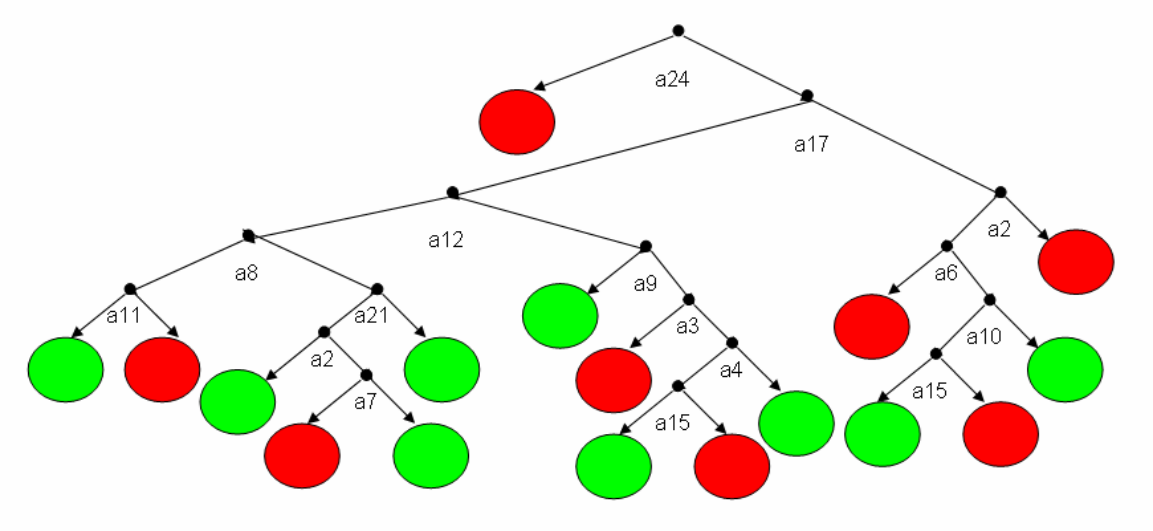

Figura 8.1: Árbol C4.5 para la extracción de reglas (Credit Scoring)

Capítulo 9

Conclusiones y Comentarios Finales

En este capítulo se resume el trabajo realizado en esta Tesis, analizando los resultados obtenidos y el cumplimiento de los objetivos planteados en el primer capítulo. Se plantea además una discusión sobre el trabajo posible a desarrollar a partir de esta Tesis y del conocimiento del estado actual del método de Support Vector Machines.

9.1. Trabajo Realizado

En esta Tesis se han mostrado los fundamentos teóricos del método Support Vector Machines, el cual muestra varias propiedades interesantes:

(1) Permite una interpretación gráfica de los conceptos teóricos como separadores lineales y no lineales, margen, Support Vectors, entre otros.

(2) Se puede utilizar la formulación Dual del problema de optimización para crear una función discriminante no lineal mediante las funciones de Kernel.

(3) Utiliza el concepto de Minimización del Riesgo Estructural a diferencia de la Minimización del Riesgo Empírico empleado por otros métodos, lo que reduce el problema de sobreajuste en los datos.

(4) Puede emplearse en bases de datos con valores fuera de rango, dado que este método utiliza sólo un subconjunto de objetos (los Support

Vectors) para construir la función discriminante, lo que evita que valores extremos determinen la solución encontrada.

(5) Es posible obtener mejores resultados en términos de efectividad (menor número de elementos mal clasificados) y eficiencia (menor tiempo requerido para seleccionar los parámetros y ejecución del modelo) en comparación a las redes neuronales.

(6) Permite el uso del modelo mismo para realizar selección de atributos (algoritmos Wrapper y Embedded), identificando los atributos irrelevantes, a diferencia de las redes neuronales.

(7) Dadas las características de SVM, es posible extenderlo a problemas de clasificación con más de dos clases (multi-class SVM [45]), Aprendizaje no Supervisado o Clustering (One Class SVM [35]) y regresión (Support Vector Regression [34])

Dadas sus características, este método es apropiado para diversas aplicaciones en el ámbito de la clasificación, como lo es la asignación de créditos y la detección del cáncer (ejemplos desarrollados en esta Tesis). Además ha sido usado en muchas otras áreas como detección de fuga, categorización de textos, reconocimiento de figuras, entre otras.

9.2. Métodos Propuestos

A pesar de las características atractivas, se mostró que SVM no realiza la tarea de selección de atributos de manera efectiva. Por esta razón, en esta Tesis se presentaron 2 metodologías para realizar la tarea de selección de atributos mediante el uso de SVM:

(1) Un método de filtro basado en sucesivas regresiones logísticas. Este enfoque consiste en la eliminación iterativa de atributos irrelevantes, los que se determinan gracias al coeficiente asociado al atributo en consideración: si el coeficiente es estadísticamente igual a cero, el atributo es candidato a ser eliminado. En cada iteración se descartará el

atributo con mayor p_valor, hasta que no se encuentren atributos irrelevantes o bien se produzca una disminución importante en la efectividad del modelo, dada por el porcentaje de elementos bien clasificados usando todo el conjunto de elementos en el entrenamiento del método.

El método presenta la ventaja de identificar de manera estadística los atributos irrelevantes y un claro criterio de parada, mostrando el momento en que la eliminación de atributos trae consecuencias negativas en la bondad de ajuste del modelo, además de evitar la eliminación de atributos relevantes pero muy correlacionados con otros atributos. La propuesta presentó buenos resultados empíricos, sin embargo la capacidad de identificar atributos irrelevantes es limitada, permitiendo la eliminación de pocos atributos.

(2) Un método Wrapper que utiliza el concepto de eliminación iterativa de atributos aplicando loo (leave-one-out) sobre los atributos. En cada iteración el método genera dos subconjuntos de los datos (uno de entrenamiento y otro para test), efectúa SVM no lineal sobre el entrenamiento para obtener los pesos de la solución (vector α) y evalúa esta solución en el subconjunto de test dejando un atributo afuera. De esta manera se recorre el conjunto de atributos y se elimina el atributo con el cual, al no ser considerado en la evaluación, se obtenga mayor efectividad en el test. El algoritmo se detiene cuando la eliminación de cualquier atributo lleve a que la solución empeore en términos de efectividad.

El método presenta ventajas similares al método anterior en el sentido de identificar claramente atributos irrelevantes y de contar con un criterio de parada. Al ser un método Wrapper presenta la ventaja adicional de usar la solución encontrada para la eliminación de atributos, lo que se traduce en un modelo más preciso, sin ajustarse demasiado a la solución gracias a que en cada iteración se generan nuevos subconjuntos de manera aleatoria. Con este método se consiguieron los mejores resultados para ambas bases de datos, y mostró ser una muy buena alternativa al método Wrapper RFE-SVM, ya que el orden del

algoritmo (y por lo tanto el esfuerzo computacional) es equivalente pero presenta mejores resultados que este último en términos de efectividad y número de atributos eliminados.

De acuerdo a la experiencia empírica, el mejor desempeño del algoritmo en términos de efectividad muchas veces se consigue antes de alcanzar el criterio de parada. Si el objetivo de la clasificación es obtener el mejor porcentaje de efectividad, se recomienda utilizar el porcentaje de acierto sobre el conjunto de test cuando se elimina un atributo como criterio complementario para decidir el número de atributos a eliminar.

9.3. Discusión y Trabajo Futuro

A partir del trabajo realizado y del conocimiento del estado actual del método SVM se proponen las siguientes áreas de trabajo e investigación:

Función de Costos y SVM: En la sección 3.2. se señala la diferencia entre los dos tipos de errores en Credit Scoring y la distinta ponderación que deberían tener, dado el costo que le representa a la entidad crediticia incurrir en ellos. En esta sección se propuso utilizar una función de costos para incorporar esta diferencia, la cual a su vez puede llevarse a SVM penalizando de manera diferente los errores en la función objetivo del primal, o bien en considerar dos cotas superiores distintas para los α_i en la formulación dual.

La formulación del problema Dual propuesta en la sección 5.5 con el parámetro $\vec{C} \in \Re = C$ se modifica a $\vec{C} \in \Re^2 = \{C^+, C^-\}$, donde el primer componente penaliza el error tipo I mientras que el segundo penaliza el error tipo II, La formulación queda de la siguiente manera:

$$\max_{\alpha} L_D \equiv \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,s=1}^{m} \alpha_i \alpha_s y_i y_s K(\overrightarrow{x_i}, \overrightarrow{x_s}) \qquad \textbf{(9.1)}$$

s.a

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C^+ \quad \forall \quad i = 1,\ldots,m \quad t.q \quad y_i = 1$$

$$0 \le \alpha_i \le C^- \quad \forall \quad i = 1,\ldots,m \quad t.q \quad y_i = -1$$

Gracias a la modificación propuesta es posible minimizar una función de costos del tipo $P_m\ C_{tipoI}\ \alpha + P_b\ C_{tipoII}$ ß variando simplemente C^+ y C^-.

En esta Tesis se consideró que ambos costos tienen la misma ponderación. Razón para esto es no perder de vista el objetivo principal de generar métodos que incorporen selección de atributos y sean aplicables a distintas variaciones de SVM como las propuestas en esta subsección. Otra razón es que no se dispone de una función de costos definida, la cual debe obtenerse mediante un trabajo en conjunto con la entidad financiera.

SVM con clases no balanceadas: Otro tema que cobra relevancia en el caso de Credit Scoring y en otras aplicaciones reales es el hecho de que las bases de datos no se encuentran balanceadas, es decir, el tamaño de las clases difiere de manera significativa. También suele considerarse un problema de balance de clases el mencionado en el punto anterior: cuando el costo de un tipo de error es muy alto con respecto al otro.

Con el fin de no perder generalidad en esta Tesis, la base de datos de Credit Scoring fue balanceada mediante un muestreo estratificado, logrando una cantidad semejante de instancias en ambas clases. La base real, sin embargo, presenta un 88% de buenos pagadores.

Para manejar bases de datos no balanceadas se sugiere el método υ-SVM con pesos (Weighted υ-SVM [43]). υ-SVM utiliza un parámetro υ para controlar la cantidad de Support Vectors y la cantidad de errores. Weighted υ -SVM incorpora además el concepto de pesos en los errores de manera similar a la modificación de la formulación Dual presentada en el punto anterior.

También es posible utilizar la modificación de la formulación Dual señalada en punto anterior donde se sugiere elegir los parámetros C^+ y C^- acorde a la siguiente regla [25]:

$$C^+ = \lambda \cdot \frac{1}{l^+}, \qquad C^- = \lambda \cdot \frac{1}{l^-} \tag{9.2}$$

donde $\lambda > 0$ y l^+ (l^-) corresponde al número de elementos de la clase positiva (negativa). De acuerdo a esto, los parámetros deben ser inversamente proporcionales al tamaño de sus clases.

Cabe señalar que los métodos de selección de atributos propuestos pueden aplicarse a estas variaciones sin requerir una modificación sustancial de los algoritmos.

Minimización norma 0 en espacio de proyección: En la sección 6.3. se señalan las dificultades que se deben enfrentar si se desea trabajar con la minimización de la norma 0 (en estricto rigor, una aproximación a esta norma dada por una función cóncava) para la selección de atributos Embedded: la necesidad de explicitar la función de proyección y el hecho que la eliminación de atributos sea en un espacio donde éstos no tienen significado (el espacio proyectado).

Pese a las dificultades señaladas es posible seguir esta directriz y desarrollar un método que combine la minimización de la norma euclidiana (SVM no lineal tradicional) con la minimización de la norma 0 para selección de atributos. Este método podría aplicarse a Kernels polinomiales simples. Finalmente, es posible establecer una relación entre los atributos seleccionados en el espacio de proyección con los atributos originales mediante una relación matemática entre ellos o bien desarrollando métodos estadísticos para encontrar la correlación existente. Se sugiere revisar [44] para mayor información.

SVM con Selección de Atributos dinámico: Resulta interesante estudiar la evolución de los atributos influyentes en el tiempo, así como la

evolución de los objetos que forman la función discriminante: los SVs. Analizando el comportamiento de estos elementos es posible detectar patrones de comportamiento en el tiempo que permitan tomar decisiones en un horizonte de tiempo mayor. Además, el estudio de los SVs permite adaptar de manera más rápida el modelo con el paso tiempo, integrando los nuevos objetos a la función de clasificación sólo cuando es necesario.

9.4. Conclusiones Finales

A continuación se muestra un cuadro resumen de los resultados obtenidos en las bases de datos Wisconsin Breast Cancer y Credit Scoring usando validación cruzada (10 particiones):

	Regresión Log.		Red MLP		SVM Kernel lineal	
	Dim	% ef.	dim	% ef.	dim	% ef.
WBC	30	93.67±2.6	30	96.66±2.1	30	94.55±2.4
Credit Scoring	49	71.11±5.1	49	70.15±5.3	49	71.1±4

	SVM Kernel poly.		SVM Kernel RBF		Filtro ACP+SVM	
	Dim	% ef.	dim	% ef.	dim	% ef.
WBC	30	96.49±2.2	30	98.25±2	20	97.72±2.2
Credit Scoring	49	75.27±3.3	49	75.54±3.6	30	71.92±4.9

	Filtro RL+SVM		RFE-SVM		Wrapper SVM	
	Dim	% ef.	dim	% ef.	dim	% ef.
WBC	26	98.07±1.9	19	97.89±1.8	18	**98.25±2**
Credit Scoring	42	74.31±4.5	25	73.97±3.8	24	**75.48±3.3**

Figura 9.1: Cuadro resumen de los resultados

A partir de estos resultados es posible obtener las siguientes conclusiones:

- SVM supera en efectividad en ambos a casos a los otros métodos de clasificación usados: la Regresión logística y la red Neuronal, alzándose como uno de lo métodos de clasificación supervisada más efectivos dentro de la Inteligencia Artificial, siendo probablemente el más efectivo de todos.

- SVM no lineal supera ampliamente en efectividad a SVM lineal en ambas aplicaciones. Con esto se demuestra de manera empírica que la flexibilidad que se logra al usar un clasificador no lineal (gracias a las funciones de Kernel) se traduce en una mayor efectividad del modelo. Si bien SVM lineal permite mayor interpretabilidad ya que los pesos están directamente asociados a los atributos (lo que facilita la tarea de la selección de atributos), vale la pena emplear funciones de Kernel y desarrollar métodos de selección de atributos para SVM no lineal ya que la efectividad lograda es mayor.

- El Kernel Gaussiano o RBF supera en efectividad en ambas aplicaciones al Kernel Polinomial, lo que es esperable debido a su naturaleza de Kernel inducido de dimensión infinita y con un parámetro ajustable (el ancho de Kernel) .

- Con respecto a la selección de atributos, ambos métodos propuestos logran un desempeño superior a los otros métodos conocidos de la literatura, cumpliendo los objetivos de la selección de atributos (disminuir la dimensionalidad del problema, lo que se traduce en ventajas como las mencionadas en el capítulo 4, sin sacrificar la efectividad del método). En especial se destaca el método Wrapper, que para la aplicación de Credit Scoring redujo la dimensionalidad en un 51% con una variación de la efectividad de apenas un 0,06%.

- Los métodos Wrapper de eliminación secuencial backward, si bien por lo general son más efectivos que los métodos de filtro, son también computacionalmente más costosos, especialmente cuando se cuenta con varias decenas de atributos y sabe sabe a priori que la mayoría son prácticamente irrelevantes. Para estos casos se sugiere un enfoque mixto, donde se utilice un método de filtro (por ejemplo análisis univariado) en una primera etapa, descartando atributos irrelevantes, y generar un subconjunto con menos variables que funcione como entrada para una segunda etapa, donde se aplique un algoritmo Wrapper enfocado en detectar los atributos relevantes para el método de

clasificación. De esta manera se combina la eficiencia de los métodos de filtro con la efectividad de los métodos Wrapper.

Como conclusión final, se cumplieron los objetivos y actividades propuestas en este trabajo: Se desarrollaron métodos efectivos para incorporar la selección de atributos en SVM no lineal, los cuales muestran un desempeño incluso mejor que los métodos tradicionales disponibles. Adicionalmente, se desarrolló un método basado en el Árbol de Decisión C4.5 que permite extraer reglas sencillas que aporten interpretabilidad al método SVM, las cuales son muy valiosas para una entidad crediticia a la hora de tomar decisiones táctico-estratégicas; se realizaron avances en el trabajo propuesto por R. Montoya [28] para la selección de atributos y SVM en su versión lineal para una posible adaptación del método al caso no lineal; se explicaron en detalle los fundamentos de SVM y sus virtudes, las cuales fueron demostradas empíricamente y se propusieron áreas de trabajo futuro a partir del trabajo realizado.

Referencias

[1] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, y J. Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. Journal of the Operational Research Society, 54(6):627–635, 2003.

[2] M. Bazaraa, H. Sherali, y C. Shetty. Nonlinear Programming: Theory and Algorithms. John Wiley, New York, second edition, 1993.

[3] A. Blum y P. Langley. Selection of relevant features and examples in machine learning. Artificial Intelligence, 97(1-2):245–271, 1997.

[4] C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (1998), no 2, 121-167.

[5] P. Bradley y O. Mangasarian, Feature Selection vía concave minimization and support vector machines, Machine Learning proceedings of the fifteenth International Conference (ICML ’98) San Francisco, California, Morgan Kaufmann, 1998, pp. 82-90.

[6] A. Cheriyadat y L. Bruce. Why Principal Component Analysis is not an Appropriate Feature Extraction Method for Hyperspectral Data, Department of Electrical and Computer Engineering, Mississippi State University, 2003

[7] C. Cortés y V. Vapnik. Support vector networks. Journal of Machine Learning, 20: 273–297, 1995.

[8] N. Cristianini y R. Holloway. Support Vector and Kernel Methods. Springer-Verlag, Berlin, Heidelberg, 2003.

[9] N. Cristianini y J. Shawe-Taylor. An Introduction to Support Vector Machines. Cambridge, UK, Cambridge University, 2000.

[10] J. Doak. An evaluation of feature selection methods and their application to computer security. Technical Report CSE-92-18, University of California, Department of Computer Science, Davis, CA, 1992.

[11] H. Drucker, D. Wu, y V. Vapnik. Support Vector Machines for Spam categorization. EEE-NN, 10(5):1048–1054, 1999.

[12] U. Fayyad. Data mining and knowledge discovery- making sense out of data. IEEE Expert-Intelligent Systems and Their Aplications, 11:20-25, 1996.

[13] R. Fletcher. Practical Methods of Optimization. John Wiley and Sons, New York, 1987.

[14] T. Van Gestel, J.A.K. Suykens, D.-E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, y J. Vandewalle. Financial time series prediction using least squares support vector machines with the evidence framework.

[15] Greene, W. H. Econometric Analysis. 5th edition. Prentice Hall, 2003

[16] I. Guyon, J. Weston, S. Barnhill, y V. Vapnik. Gene selection for cancer classification using support vector machines. Machine Learning, 46:389--422, Kluwer Academic Publishers, Boston, 2000.

[17] G. John, R. Kohavi, y K. Pfleger. Irrelevant features and the subset selection problem. In 11th Int. Conf. on Machine Learning, pages 121–129, New Brunswick, NJ, 1994.Morgan Kaufmann.

[18] D. Jonson, Aplied multivariate methods for data analysts, Brooks Cole Publishing company, New York, 1998.

[19] W. Kim, B. Choi, E.-K. Hong, y S.-K. Kim. A taxonomy of dirty data. Data Mining and Knowledge Discovery, 7:81–99, 2003.

[20] P. Lachenbruch. An almost unbiased method of obtaining confidence intervals for the probability of misclassification in discriminant analysis. Biometrics, pp. 639–645, 1967.

[21] P. Langley. Selection of relevant features in machine learning. in AAAI Fall Symposium on Relevance, pp. 140–144, 1994.

[22] H. Liu y H. Motoda. Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, London, UK, 1998.

[23] H. Liu y L. Yu. Toward integrating feature selection algorithms for classification and clustering. IEEE Trans. on Knowledge and Data Eng., 17(3):1–12, 2005.

[24] C. Lu, T. Van Gestel, J.A.K. Suykens, S. Van Huffel, I. Vergote, y D. Timmerman. Preoperative prediction of malignancy of ovarium tumor using least squares support vector machines. Artificial Intelligence in Medicine, 28(3):281–306, 1999.

[25] F. Markowetz, Support Vector Machines in Bioinformatics, Master's thesis, University of Heidelberg, 2001.

[26] D. Martens, B. Baesens, T. Van Gestel y J. Vanthienen. Comprehensible Credit Scoring Models using Rule Extraction from Support Vector Machines. European Journal of Operational Research, 2006.

[27] J. Miranda. Modelo de predicción de fugas voluntarias para una institución financiera utilizando Support Vector Machines. Trabajo de Tesis para optar al grado de Magister en Gestión de Operaciones, 2006.

[28] R. Montoya y R. Weber. Support Vector Machines penalizado para selección de atributos. XI CLAIO, Concepción, Chile, 27-30 de octubre de 2002

[29] R. Montoya. Programación matemática para Data Mining utilizando SVM para selección de atributos. Trabajo de Tesis para optar al grado de Magister en Gestión de Operaciones, 2002.

[30] P. Ñarendra y K. Fukunaga. A branch and bound algorithm for feature subset selection.IEEE Transactions on Computer, C-26(9):917–922, 1977.

[31] D. Pyle. Data preparation for data mining. Morgan Kaufmann Publishers, 1999.

[32] J.R. Quinlan. C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA,1993.

[33] J. Schlimmer. Efficiently inducing determinations: A complete and efficient search algorithm tat uses optimal pruning. In 10th Int. Conf. on Machine Learning, pages 284–290, Amherst, MA, 1993. Morgan Kaufmann.

[34] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. NeuroCOLT2 Technical Report NC2-TR-1998-030, 1998.

[35] B. Schölkopf , A. J. Smola , J. Platt , J. Shawe-Taylor y R. C. Williamson. Estimating the support of a high-dimensional distribution. Microsoft Research Technical Report MSR-TR-99-87, 1999.

[36] B. Schölkopf, K. Sung, C. Burgues, J.C. Girosi, P. Niyogui, y V. Vapnik. Comparing Support Vector Machine with Gaussian Kernels to Radial Basis Function classifiers. IEEE Transaction on Signal Processing, 45(11):2758–2765, 1997.

[37] R. Setiono y H. Liu. Neural–network feature selector. IEEE Trans. on Neural Networks, 8(3):654–662, 1997

[38] F. Tay y L. Cao, A comparative study of saliency analysis and genetic algorithm for feature selección in support vector machines, Intelligent Data Analysis 5 (2001), 191-209.

[39] V. Vapnik. Estimation of dependences based on empirical data. Springer-Verlag, New York, 1982.

[40] V. Vapnik. The nature of statistical learning theory. Springer–Verlag, New York, 1995.

[41] V. Vapnik. Statistical Learning Theory. John Wiley and Sons, New York, 1998.

[42] G. Wahba. An introduction to model building with reproducing Kernel Hilbert spaces, Tech. Report 1020, Statistics Department, University of Wisconsin, Madison, Wisconsin, 2000.

[43] M. Wang, J. Yang, G. Liu, Z. Xu and K. Chou. Weighted-support vector machines for predicting membrane protein types based on pseudo-amino acid composition. Protein Engineering Design and Selection 17(6):509-516, 2004.

[44] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. The use of zero-norm with linear models and kernel methods. JMLR special Issue on Variable and Feature Selection, 3:1439 -- 1461, 2002.

[45] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, Egham, TW20 0EX, UK, 1998.

[46] J. Yang, G. Liu. The evaluation of classification models for credit scoring. Arbeitsbericht Nr. 02/2002 Institut für Wirtschaftsinformatik, Georg-August-Universität Göttingen.

[47] S. Zhang, C. Zhang, y Q. Yang. Data preparation for data mining. Applied Artificial Intelligence, 17(5–6):375–381, 2003.

[48] S. Zilberstein. Using anytime algorithms in intelligence systems. AI magazine, Fall:73–83, 1996.

Capítulo 10

Anexos

10.1. Tablas de resultados de los modelos

10.1.1. Wisconsin Breast Cancer

Validación SVM Kernel Lineal	
Tupla	**Acierto(%)**
1	89,47
2	92,98
3	94,74
4	92,98
5	94,74
6	94,74
7	94,74
8	96,49
9	98,25
10	96,43
Promedio	94,55
Varianza	5,76

Tabla 10.1: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Kernel lineal, mejor configuración: C=100

Validación SVM Kernel Polinomial	
Tupla	**Acierto(%)**
1	92,982
2	96,491
3	96,491
4	100
5	96,491
6	96,491
7	96,491
8	92,982
9	98,246
10	98,214
Promedio	96,49
Varianza	4,78

Tabla 10.2: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Kernel Polinomial, mejor configuración d=3, C=100.

Validación SVM Kernel RBF	
Tupla	**Acierto(%)**
1	96,491
2	96,491
3	94,737
4	100
5	96,491
6	100
7	100
8	98,246
9	100
10	100
Promedio	98,25
Varianza	4,10

Tabla 10.3: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Kernel RBF, mejor configuración ρ=2, C=100.

Validación SVM Filtro ACP	
Tupla	**Acierto(%)**
1	94,737
2	96,491
3	94,737
4	100
5	96,491
6	100
7	100
8	96,491
9	100
10	98,214
Promedio	97,72
Varianza	4,82

Tabla 10.4: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Filtro ACP, Kernel RBF, mejor configuración ρ=2, C=80, 20 atributos.

Validación SVM Filtro LR	
Tupla	**Acierto(%)**
1	96,491
2	96,491
3	94,737
4	98,246
5	96,491
6	100
7	100
8	98,246
9	100
10	100
Promedio	98,07
Varianza	3,73

Tabla 10.5: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Filtro LR, Kernel RBF, mejor configuración ρ=2, C=40, 26 atributos

Validación RFE-SVM	
Tupla	**Acierto(%)**
1	96,491
2	96,491
3	94,737
4	98,246
5	96,491
6	98,246
7	100
8	100
9	98,246
10	100
Promedio	97,89
Varianza	3,28

Tabla 10.6: Porcentaje de acierto sobre el conjunto de validación para el modelo RFE-SVM, Kernel RBF, mejor configuración ρ=3, C=100, 19 atributos.

Validación Wrapper SVM	
Tupla	**Acierto(%)**
1	96,491
2	96,491
3	94,737
4	100
5	96,491
6	100
7	100
8	98,246
9	100
10	100
Promedio	98,25
Varianza	4,10

Tabla 10.7: Porcentaje de acierto sobre el conjunto de validación para el modelo Wrapper de elaboración personal, Kernel RBF, mejor configuración ρ=2, C=100, 18 atributos.

10.1.2. Credit Scoring

Validación SVM Kernel Lineal	
Tupla	**Acierto(%)**
1	68,71
2	78,91
3	73,47
4	70,75
5	68,49
6	73,97
7	67,12
8	71,23
9	73,29
10	65,07
Promedio	71,10
Varianza	16,06

Tabla 10.8: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Kernel lineal, mejor configuración: C=30

Validación SVM Kernel Polinomial	
Tupla	**Acierto(%)**
1	75,51
2	78,91
3	78,91
4	75,51
5	76,03
6	73,29
7	71,23
8	73,97
9	79,45
10	69,86
Promedio	75,27
Varianza	10,65

Tabla 10.9: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Kernel Polinomial, mejor configuración d=2, C=0,05.

Validación SVM Kernel RBF	
Tupla	**Acierto(%)**
1	75,51
2	77,55
3	78,91
4	79,59
5	76,03
6	76,03
7	69,18
8	71,92
9	79,45
10	71,23
Promedio	75,54
Varianza	13,30

Tabla 10.10: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Kernel RBF, mejor configuración ρ=4, C=30.

Validación SVM Filtro ACP	
Tupla	**Acierto(%)**
1	77,55
2	68,03
3	78,91
4	76,87
5	67,12
6	71,92
7	71,23
8	67,81
9	74,66
10	65,07
Promedio	71,92
Varianza	23,89

Tabla 10.11: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Filtro ACP, Kernel RBF, mejor configuración ρ=4, C=70, 30 atributos.

Validación SVM Filtro LR	
Tupla	**Acierto(%)**
1	74,15
2	76,19
3	83,67
4	71,43
5	73,29
6	76,71
7	69,18
8	71,23
9	78,08
10	69,18
Promedio	74,31
Varianza	20,29

Tabla 10.12: Porcentaje de acierto sobre el conjunto de validación para el modelo SVM, Filtro LR, Kernel RBF, mejor configuración ρ=5, C=30, 42 atributos.

Validación RFE-SVM	
Tupla	**Acierto(%)**
1	71,43
2	76,19
3	79,59
4	75,51
5	72,6
6	75,34
7	69,86
8	72,6
9	78,77
10	67,81
Promedio	73,97
Varianza	14,32

Tabla 10.13: Porcentaje de acierto sobre el conjunto de validación para el modelo RFE-SVM, Kernel RBF, mejor configuración ρ=3, C=40, 25 atributos.

Validación Wrapper SVM	
Tupla	**Acierto(%)**
1	72,79
2	74,83
3	82,99
4	74,83
5	73,97
6	78,08
7	72,6
8	77,4
9	75,34
10	71,92
Promedio	75,48
Varianza	10,91

Tabla 10.14: Porcentaje de acierto sobre el conjunto de validación para el modelo Wrapper SVM de elaboración personal, Kernel RBF, mejor configuración ρ=2, C=60, 24 atributos.

10.2. Resultados otros métodos

10.2.1. Wisconsin Breast Cancer

Regresion Logística

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	533	93.6731 %
Incorrectly Classified Instances	36	6.3269 %
Kappa statistic	0.8654	
Mean absolute error	0.0642	
Root mean squared error	0.2517	
Relative absolute error	13.7324 %	
Root relative squared error	52.0545 %	
Total Number of Instances	569	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.929	0.059	0.904	0.929	0.916	a
0.941	0.071	0.957	0.941	0.949	b

=== Confusion Matrix ===

a	b	<-- classified as
197	15	a = a
21	336	b = b

Red Neuronal MLP

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	550	96.6608 %
Incorrectly Classified Instances	19	3.3392 %
Kappa statistic	0.9284	
Mean absolute error	0.0351	
Root mean squared error	0.17	
Relative absolute error	7.5138 %	
Root relative squared error	35.158 %	
Total Number of Instances	569	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.948	0.022	0.962	0.948	0.955	a
0.978	0.052	0.969	0.978	0.974	b

=== Confusion Matrix ===

a	b	<-- classified as
201	11	a = a
8	349	b = b

10.2.2. Credit Scoring

Regresion Logística

Correctly Classified Instances	1041	71.1066 %
Incorrectly Classified Instances	423	28.8934 %
Kappa statistic	0.4184	
Mean absolute error	0.361	
Root mean squared error	0.4358	
Relative absolute error	72.3698 %	
Root relative squared error	87.258 %	
Total Number of Instances	1464	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.651	0.235	0.716	0.651	0.682	b
0.765	0.349	0.707	0.765	0.735	a

=== Confusion Matrix ===

a	b	<-- classified as
454	243	a = b
180	587	b = a

Red Neuronal MLP

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	1027	70.1503 %
Incorrectly Classified Instances	437	29.8497 %
Kappa statistic	0.4004	
Mean absolute error	0.301	
Root mean squared error	0.5187	
Relative absolute error	60.3339 %	
Root relative squared error	103.8495 %	
Total Number of Instances	1464	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.664	0.265	0.695	0.664	0.679	b
0.735	0.336	0.707	0.735	0.721	a

=== Confusion Matrix ===

a	b	<-- classified as
463	234	a = b
203	564	b = a

10.3. Resultado Árbol C4.5

```
a24 <= 0
|   a17 <= 0.78
|   |   a2 <= 0.27: b (218.0/27.0)
|   |   a2 > 0.27
|   |   |   a6 <= 0.12
|   |   |   |   a10 <= 0.11: a (90.0/15.0)
|   |   |   |   a10 > 0.11
|   |   |   |   |   a15 <= 0: b (13.0)
|   |   |   |   |   a15 > 0: a (17.0/7.0)
|   |   |   a6 > 0.12: b (106.0/32.0)
|   a17 > 0.78
|   |   a12 <= 0
|   |   |   a9 <= 0.55
|   |   |   |   a3 <= 0.33
|   |   |   |   |   a4 <= 0.22: a (91.0/20.0)
|   |   |   |   |   a4 > 0.22
|   |   |   |   |   |   a15 <= 0: b (18.0/2.0)
|   |   |   |   |   |   a15 > 0: a (13.0/4.0)
|   |   |   |   a3 > 0.33: b (125.0/32.0)
|   |   |   a9 > 0.55: a (90.0/13.0)
|   |   a12 > 0
|   |   |   a8 <= 0.86
|   |   |   |   a21 <= 0.1: a (463.0/59.0)
|   |   |   |   a21 > 0.1
|   |   |   |   |   a2 <= 0.18
|   |   |   |   |   |   a7 <= 0.7: a (19.0/6.0)
|   |   |   |   |   |   a7 > 0.7: b (28.0/2.0)
|   |   |   |   |   a2 > 0.18: a (50.0/11.0)
|   |   |   a8 > 0.86
|   |   |   |   a11 <= 0.13: b (17.0/2.0)
|   |   |   |   a11 > 0.13: a (14.0/5.0)
a24 > 0: b (92.0/4.0)
```