UNIVERSIDAD SIMÓN BOLÍVAR

DECANATO DE ESTUDIOS DE POSTGRADO

COORDINACIÓN DE POSTGRADO EN CÓMPUTO CIENTÍFICO

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

TRABAJO DE GRADO

ALGORITMO DE PUNTOS INTERIORES DE BAJO COSTO APLICADO AL PROBLEMA DE MÁQUINAS DE SOPORTE VECTORIAL

Por

Esnil Josué Guevara Mendoza

Diciembre, 2013

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CÓMPUTO CIENTÍFICO
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

ALGORITMO DE PUNTOS INTERIORES DE BAJO COSTO APLICADO AL PROBLEMA DE MÁQUINAS DE SOPORTE VECTORIAL

Trabajo de Grado presentado a la Universidad Simón Bolívar por

Esnil Josué Guevara Mendoza

Como requisito parcial para optar al grado de

Magister en Ciencias de la Computación
Con la asesoría de la profesora

María de los Ángeles González

Diciembre, 2013

Aprobada por:

Jurado	Fecha
Prof. Nelson Hernández	
Jurado	Fecha
Prof. Luis Rodríguez	
Tutor	Fecha
Prof. Víctor Griffin	
Director del Departamento	Fecha
Prof. José Marcano	

A Dios, porque en los momentos donde no podia avanzar, El me lleno de su sabiduria.

AGRADECIMIENTOS

Al Señor Dios Todopoderoso, porque Él le dió un propósito a mi vida.

A mi padre, por todo el amor y la ayuda incondicional que me ha dado para seguir adelante, y por los abrazos que me da cada mañana.

A mi madre, porque gracias a ella pude conocer el temor de Dios en mi vida, sus regaños y consejos me han enseñado a crecer en todas las áreas de mi vida.

A mis hermanos, por estar conmigo cuando los necesito; te quiero Jesús y te quiero Ruth.

A mis familiares, viejos amigos, y a quienes recién se sumaron a mi vida para hacerme compañia con sus sonrisas de ánimo, en especial a la nina que ha sido una gran amiga.

A mis hermanos de la iglesia por apoyarme con sus oraciones en mis estudios y en el ministerio de alabanza Luz y Vida.

A mis profesores de la Universidad de Carabobo, del departamento de Matemática en la Facultad de Ciencia y Tecnología, porque en sus aulas recibí el conocimiento intelectual y humano para desenvolverme en el área profesional.

A Fernando Cedeño, Anthony Cho, Marlyn Cuadrado, Freddy Narea y Patricia Chirivella por animarme y convercerme de hacer mis estudios de maestria.

A mis amigos de trabajo del departamento de planificación de Laboratorios Vargas, por estar pendiente de la entrega de mi trabajo especial de grado.

Especial agradecimiento a mi tutora María de los Angeles Lima-González, por estar siempre pendiente de mis avances y por creer que yo lo podia hacer.

A todos ellos, Gracias!

UNIVERSIDAD SIMÓN BOLÍVAR

DECANATO DE ESTUDIOS DE POSTGRADO

COORDINACIÓN DE POSTGRADO EN CÓMPUTO CIENTÍFICO

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

ALGORITMO DE PUNTOS INTERIORES DE BAJO COSTO APLICADO AL PROBLEMA DE MÁQUINAS DE SOPORTE VECTORIAL

Estudiante: Guevara Esnil
Carnet: 0987442
Diciembre 2013

RESUMEN

En esta investigación se propone un algoritmo de bajo costo, primal-dual de puntos interiores, para resolver el problema cuadrático, convexo, con restricciones tipo knapsack continuas, donde el Hessiano de la función objetivo es grande, denso y mal condicionado. La idea subyacente en la propuesta es considerar una aproximación de la matriz Hessiano por un múltiplo de la identidad que usa la longitud espectral. Se aplica el algoritmo propuesto al problema que surge en las Máquinas de Vectores de Soporte (MVS) en dos formas, una sin el paso predictor y otro con identificación de variables. Ambos resultados muestran que el método propuesto es eficaz y permite resolver problemas reales donde el método tradicional predictor-corrector falla, y que además reduce sustancialmente el tiempo computacional.

Palabras Claves: Máquina de soporte vectorial, vectores de soporte, método primal-dual de punto interior, método de escalamiento afín ASL.

TABLA DE CONTENIDO

pagina

LISTA DE TABLAS

Tabla pagina

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

MVS	Maquinas de Vectores de Soporte.
QP	Quadratic Programming.
KKT	Karush-Kuhn-Tucker.
OSH	Optimal Separating Hyperplane.
MPI	Métodos de Punto Interior.
OCR	Optical Character Recognition.
ADN	Ácido Desoxirribonucléico.
ASL	Affine-scaling Algorithm.
SMO	Secuential Minimal Optimization.

LISTA DE SIMBOLOS

σ	Parámetros de centrado.
μ	Medida de dualidad.
$\mathfrak{C}$	Conjunto de restricciones asosiados al problema cuadrático.
C	Parámetro de regularización.
$\mathscr{C}$	Camino central.

INTRODUCCIÓN

Las Máquinas de Soporte Vectorial o Máquinas de Vectores de Soporte MVS (conocidas en la literatura por SVM por sus siglas en inglés "Support Vector Machines") son algoritmos de aprendizaje automático que permiten resolver problemas de clasificación y regresión de manera muy eficiente y que se han posicionado por encima de otras técnicas, tales como las redes neuronales. Las MVS están siendo utilizadas con éxito en diferentes áreas de la informática e inteligencia artificial. Actualmente hay muchas aplicaciones [24] que utilizan las técnicas de las MVS como por ejemplo las de OCR ("Optical Character Recognition") por la facilidad de las MVS de trabajar con imágenes como datos de entrada, entre ellas el reconocimiento de firmas [15], reconocimientos de rostros [28] y categorización de textos [16], también se pueden usar en motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica.

Las MVS se basan en la teoría del aprendizaje estadístico desarrollada por V. Vapnik y A. Chervonenkis (1998) [33]. El éxito de las MVS ra-dica en tres ventajas fundamentales: la primera consiste en que poseen una sólida funtamentación matemática. La segunda consiste en que se basan en el concepto de minimización del riesgo extructural, esto es, minimizar la probabilidad de una clasificación errónea sobre nuevos ejemplos, particularmente importante cuando se dispone de pocos datos de entrenamiento. La tercera ventaja radica en que dispone de potentes herramientas y algoritmos para hallar la solución de manera rápida y eficiente. Gracias a

esto, se han creado aplicaciones para la solución de problemas reales, destacándose como una herramienta robusta en problemas complejos, ruidosos y con escasos datos.

Para resolver problemas tipo MVS (entrenar una función que sea capaz de identificar patrones en los datos que permitan hacer predicciones útiles sobre observaciones nuevas), es necesario resolver un problema de optimización cuadrático convexo. Las principales técnicas usadas para resolver este tipo de problemas son: gradiente estocástico ascendente, método de Newton, gradiente conjugado descendente, y el método primal-dual de puntos interiores. Sin embargo, los métodos tradicionales para resolver problemas cuadráticos y convexos tienen dificultades para resolver los problemas MVS por la densidad, el tamaño y el mal condicionamiento del Hessiano de la función objetivo.

Los métodos primal-dual de punto interior resuelven problemas de optimización cuadrática, aplicando variantes del método de Newton, luego modificando las direcciones de búsqueda y tamaño de paso, de manera que la restricción de no negatividad de las variables se satisfagan estrictamente en cada iteración. Sin embargo la mayor dificultad de estos algoritmos es la resolución por iteración de un sistema lineal para determinar una dirección de Newton. Este es el aspecto de mayor costo computacional y el que determina la estabilidad y robustez de los algoritmos. Entre las alternativas más llamativas para resolver estos sistemas son: la ecuación normal y el sistema aumentado. Dadas las características que presenta la matriz Hessiana asociada a la función cuadrática de las MVS, se propone en este trabajo aproximarla por la matriz identidad multiplicada por λ_k la cual cambia en cada iteración y donde λ_k es la longitud espectral introducida en [13]. Con esta aproximación, se puede explotar la esparcidad de la matriz involucrada en el sistema lineal a resolver para obtener la dirección de búsqueda y construir el algoritmo de puntos interiores de bajo costo.

Partiendo con la comprensión del método predictor-corrector primal dual de puntos interiores (para una descripción referimos a [27]), se comenzará con programar el código en MATLAB [25], y luego de generar el método propuesto, se optimizará el código de la mejor manera para disminuir el tiempo de corrida, pero aun tratándose de un lenguaje de alto nivel, el tiempo de procesamiento serguirá siendo muy elevado, lo que limita el número de muestras con las que se pueden trabajar.

La mejora en el código se traduce en una mayor velocidad de ejecución, lo que conlleva un aumento del número de muestras que se pueden entrenar/clasificar. Para lograr esto se ha optado por reprogramar el código en un lenguaje de más bajo nivel. El lenguaje elegido ha sido C++, que aunque es más complejo, presenta mejor redimiento que MATLAB en problemas grandes.

Este código en C++ se ha introducido mediante el empleo de funciones MEX, que permiten que una función en C++ sea llamada desde un entorno MATLAB. Además, se han creado una serie de scripts y funciones en MATLAB que se describirán en los capítulos posteriores.

El documento está dividido en siete capítulos. En el primero se exponen tanto el planteamiento del problema como los objetivos del proyecto y su importancia. En el segundo capítulo, veremos el marco teórico en el que se desarrolla el problema y cuyo conocimiento resultará fundamental para la posterior comprensión de la solución propuesta. En el tercer capítulo trataremos todo lo referente a las Máquinas de Vectores de Soporte, tales como su formulación y algunas enfoques que se usan en la práctica para resolverlas. El cuarto capítulo desglosa el desarrollo metodológico, donde se derivará el algoritmo de bajo costo primal-dual de puntos interiores que se usará para resolver los problemas reales. En el quinto capítulo se dará una breve explicación del diseño del software, y como se ejecuta la misma en MATLAB y el terminal de Ubuntu. En el sexto capítulo se mostrarán los resultados en base a

un conjunto de datos seleccionados, donde se comparan el error de generalización, el valor de la función objetivo, el número de vectores de soporte y los tiempos de ejecución entre el algoritmo de Bajo Costo, ASL y LIBSVM. Por último se expondrán las conclusiones extraídas en la realización del presente trabajo de investigación.

CAPÍTULO I

EL PROBLEMA

1.1 Planteamiento del Problema

La teoría de las Máquinas de Vectores de Soporte (MVS), se basa en la teoría del aprendizaje estadístico desarrollada por V. Vapnik y A. Chervonenkis [33]. Las MVS son métodos que permiten clasificar conjuntos de datos pertenecientes a dos clases distintas. Para hallar la frontera de separación de un conjunto de datos, es necesario resolver un problema de programación cuadrática [8] del siguiente tipo:

$$\begin{aligned} \underset{x}{\text{minimizar}} \quad & \frac{1}{2}x^tQx - e^tx \\ \text{sujeto a} \quad & x^ty = 0 \\ & 0 \leq x_i \leq C,\ i = 1, \ldots, n. \end{aligned} \tag{1.1}$$

Este es un problema de optimización cuadrático convexo con una restricción de igualdad y restricciones de cotas, donde Q (o Hessiano) es una matriz de orden $n \times n$ positiva semidefinida tal que $Q_{ij} = y_iy_jK(z_i, z_j)$, los valores (y_i, z_i) para $i = 1, ..., n$, forman el conjunto de los datos de entrenamiento, los vectores $x \in \mathbb{R}^n$, $y_i \in$ {-1,1} y $z_i \in \mathbb{R}^m$; $K : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$ denota la función Kernel, el parámetro de regularización C está asociado al margen de error en la MVS y e es un vector de unos.

Un Kernel K es una función, tal que para todo x,$y \in \mathbb{R}^m$

$$K(x, y) \quad = \quad \phi(x)^t\phi(y)$$

donde ϕ es una función que mapea los datos del espacio de entrada $\mathbb{R}^m$ a un espacio de mayor dimensión $\mathbb{R}^f$ $(f \gg m)$.

La representación del producto interno mediante la función Kernel requiere de multiples cálculos y la matriz Hessiano de (1.1) puede tornarse singular (mal condicionada) al tener muchos atributos poco relevantes o con valores muy pequeños.

Los métodos tradicionales para resolver problemas de programación cuadrática, tienen dificultades para resolver (1.1) por la densidad, el tamaño y el mal condicionamiento del Hessiano de la función objetivo. Por lo cual los métodos mas usados para resolver este problema de programación cuadrática, combinan una reducción del problema de optimización y algoritmos de optimización de bajo costo que explotan la estructura del problema (1.1) para resolver, en forma eficiente, los subproblemas (cuadráticos y convexos) que se forman. Recientemente, María D. González-Lima, William W. Hager, y Hongchao Zhang [13], presentaron el algoritmo ASL que compara favorablemente con alternativas usadas como LIBSVM [10], SOM [21, 29], GPDT [31, 32, 36], SVM^{light} [17, 18, 22]. ASL se basa en la linealización de las condiciones de optimalidad de primer orden en la variable x y se aproxima el Hessiano de la función objetivo en (1.1) por un múltiplo positivo de la matriz de identidad. Esto condujo a una ecuación no lineal que debe resolverse en cada iteración para encontrar la dirección de búsqueda d_k, y su multiplicador asociado μ_k. El múltiplo usado en la ASL en cada iteración del método es la longitud espectral [3] denotado λ_k, donde λ_k es la solución del problema $\min_{\lambda \geq \lambda_0} \|\lambda(x_k - x_{k-1}) - (\nabla f(x_k) - \nabla f(x_{k-1}))\|_2$. La motivación de esta elección es que proporciona una aproximación a dos puntos a la ecuación de la secante subyacente a los métodos Quasi-Newton. Debido a sus simplicidad, eficacia, y bajo requerimiento de memoria, ha sido utilizado en muchos métodos [5, 12, 23, 32].

Otras alternativas que resuelven problemas de programación cuadrática son los métodos primal-dual de punto interior, pero la dificultad que presentan estos algoritmos es la resolución por iteración de un sistema de ecuaciones lineales para determinar la dirección de búsqueda. Resolver este sistema de ecuaciones (que requiere el almacenamiento explícito de la matriz Q) es el aspecto de mayor costo computacional y el que determina la estabilidad y robustez del algoritmo.

Esta investigación propone un algoritmo de bajo costo basado en los métodos de punto interior (predictor-corrector Mehrotra [26]) con aplicación a las MVS, donde se aproxime la matriz Hessiano por una matriz identidad multiplicada por λ_k o longitud espectral el cual se obtiene siguiendo la idea en [13]. Con esta aproximación, se puede explotar la esparcidad de la matriz y la ventaja que tiene en relación al algoritmo presentado en ASL [13] es que las variables se actualizan en cada iteración sin necesidad de resolver una ecuación no lineal.

1.2 Objetivos

1.2.1 Objetivo General

Proponer un método de bajo costo (que no use información de segundo orden) basado en los métodos primal-dual de puntos interiores para resolver el problema de programación cuadrático convexo que surge en las Máquinas de Vectores de Soporte (MVS).

1.2.2 Objetivos Específicos

- Estudiar el método primal-dual de puntos interiores para programación cuadrática.
- Derivar el algoritmo de bajo costo basado en los métodos primal-dual de puntos interiores para el problema de optimización cuadrática sujeta a una restricción de igualdad y restricciones de cotas.

- Programar el algoritmo propuesto en MATLAB y analizar su comportamiento en problemas artificiales del tipo MVS usando los Kernels: Lineal, Polinomial y Guassiano.
- Ajustar los parámetros del método propuesto.
- Comprobar el método propuesto bajo la plataforma de lenguaje C/C++ corriendo con problemas de la vida real que provee la librería LIBSVM.
- Comparar con otros métodos existentes (ASL y LIBSVM).

1.3 Justificación e Importancia

En muchas aplicaciones de la vida real la dimensión de los datos de entrenamiento llegan a ser muy grandes, por lo que en la fase de entrenamiento de los algoritmos MVS se originan problemas de programación cuadráticas desafiantes. Los métodos para resolver estos problemas, se basan en la descomposición del problema en subproblemas más pequeños, pero aún siendo de menor dimensión suponen un desafío su resolución eficiente. En esta investigación se propone un algoritmo de bajo costo como una nueva alternativa para resolver los problemas de programación cuadrática que surgen de las MVS. Aquí no se emplearán técnicas de descomposición, sino que se considerarán problemas de mediana escala. La intención a futuro, pero que se escapa de este trabajo de investigación, es combinar esta metodología con técnicas de descomposición. En esta propuesta, se tratará de resolver problemas de mediana escala mediante un algoritmo que surge de los métodos primal-dual de puntos interiores [27], al considerar una sustitución de la matriz Hessiano por una matriz diagonal que sea múltiplo de la identidad, y donde ese múltiplo se consigue a través de la longitud espectral [3].

CAPÍTULO II
MARCO TEÓRICO DE LA INVESTIGACIÓN

En el siguiente apartado vamos a explicar tanto conceptos básicos de optimización, como herramientas para convertir un problema a su correspondiente problema dual, condiciones de Karush-Kuhn-Tucker y condiciones necesarias y suficientes para determinar la solución de un problema de optimización.

2.1 Optimización

La teoría de optimización [27] es una herramienta imprescindible en el desarrollo de técnicas usadas por las Máquinas de Vectores de Soporte [8], ésta es la razón por la que se justifica la presencia de este capítulo.

A continuación se definirán los conceptos de conjunto convexo, función convexa, problema de optimización dual y condiciones KKT. Un problema de optimización trata de encontrar el máximo o el mínimo de una función sujeta a una serie de restricciones que pueden ser de igualdad o desigualdad. Dependiendo de la función a optimizar y de las restricciones, tenemos infinidad de tipos de problemas de optimización en los que podemos usar diferentes algoritmos. En este capítulo vamos a centrarnos en los problemas de optimización convexa cuadráticos con restricciones lineales, es decir aquellos en los que la función objetivo es cuadrática y las restricciones son lineales.

2.1.1 Problemas de Optimización

En optimización sin restricciones se minimiza una función objetivo que depende de variables reales sin restricciones sobre los valores de esas variables. La formulación matemática es:

$$\underset{x\in\Omega}{\text{minimizar}} \quad f(x) \tag{2.1}$$

donde $x \in \mathbb{R}^n$ y f es una función continua $f : \mathbb{R}^n \to \mathbb{R}$.

Definición 1. *Un conjunto $\Omega \subseteq \mathbb{R}^n$ es convexo si para cualquier punto $x, y \in \Omega$ y $\theta \in$ [0,1] se verifica:*

$$\theta x + (1-\theta)y \in \Omega$$

Definición 2. *Sea $\Omega \subseteq \mathbb{R}^n$ convexo, una función f, definida en $f : \Omega \to \mathbb{R}$ es convexa si para todo $x, y \in \Omega$ existe θ tal que:*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

Definición 3. *(Mínimo local). Un mínimo $x^* \in \Omega$ de f es local, si existe ϵ tal que x^*, para todo $x \in (x^* - \epsilon, x^* + \epsilon)$, $f(x) \geq f(x^*)$.*

Definición 4. *(Mínimo global). Un mínimo global x^* de f es global en un dominio Ω, si para todo $x \in \Omega$ y $x^* \in \Omega$, $f(x) \geq f(x^*)$.*

Suponga que $f(x)$ es al menos continua dos veces diferenciable ($f \in C^2$).

Definición 5. *El gradiente de una función $f : \mathbb{R}^n \to \mathbb{R}$ se escribirá en este trabajo como:*

$$\nabla f(x)$$

donde $\nabla f(x)$ denota el vector de la primera derivada parcial, cuyo i-ésimo componente es $\partial f(x)/\partial x_i$.

Definición 6. *El Hessiano de una función* $f : \mathbb{R}^n \to \mathbb{R}$ *es una matriz de* $n \times n$ *cuyo i,j-ésima componente es la segunda derivada parcial* $\partial^2 f(x)/\partial x_i \partial x_j$ *y se denotará en este trabajo como:*

$$Q = \nabla^2 f(x)$$

(NOTA: Q es siempre simétrica).

Las condiciones necesarias de optimalidad para el caso de optimización sin restricciones, es decir cuando $\Omega = \mathbb{R}^n$ son derivadas asumiendo que un punto x^* es un mínimo local y viendo que ocurre con $\nabla f(x^*)$ y $\nabla^2 f(x^*)$. Las pruebas de estos teoremas se pueden encontrar en [14] y [27].

Teorema 1. *(Condiciones necesarias de primer orden). Si* x^* *es un minimizador local de* $f(x)$ *y* f *es continuamente diferenciable en una vecindad abierta de* x^**, entonces* $\nabla f(x^*) = 0$.

Teorema 2. *(Condiciones necesarias de segundo orden). Si* x^* *es un minimizador local de* $f(x)$ *y* $\nabla^2 f(x)$ *es continua en una vecindad abierta de* x^**, entonces* $\nabla f(x^*) = 0$ *y* $\nabla^2 f(x^*)$ *es positiva semidefinida.*

Supongamos que nosotros encontramos un punto que satisface las condiciones antes descritas. Nosotros podemos garantizar que este es un minimizador, siempre que se cumplan la condiciones suficientes de optimalidad de segundo orden.

Teorema 3. *(Condiciones suficientes de segundo orden). Supongamos que* $\nabla^2 f$ *es continua en una vecindad abierta de* x^**,* $\nabla f(x^*) = 0$ *y* $\nabla^2 f(x^*)$ *es positiva definida. Entonces* x^* *es un único minimizador local de* $f(x)$.

Definición 7. *Si una función es convexa y diferenciable, y además el gradiente es cero en un punto* x^**, entonces existe un mínimo de la función en ese punto. En*

otras palabras:

$$\nabla f(x^*) = 0$$

junto con la convexidad de f es condición suficiente para que x^* sea un mínimo.

2.1.2 Problema de Optimización con Restricciones Lineales

Definición 8. *(Problema de optimización). Dadas las funciones convexas f, g_i, $i = 1, \cdots, k$, y h_i, $i = 1, \cdots, m$, definidas en un dominio $\Omega \subseteq \mathbb{R}^n$. El problema de optimización será el siguiente:*

$$\begin{aligned} \underset{x\in\Omega}{\textit{minimizar}} \quad & f(x) \\ \textit{sujeto a} \quad & g_i(x) \leq 0,\ i = 1, ..., k \\ & h_i(x) = 0,\ i = 1, ..., m \end{aligned} \tag{2.2}$$

donde $f(x)$ es conocida como función objetivo, $h_i(x)$ son las restricciones de igualdad y $g_i(x)$ las restricciones de desigualdad.

La solución del problema de optimización convexa (2.2) viene dada por $x^* \in \mathfrak{C}$ donde $\mathfrak{C}$ se conoce como región factible y se define de la siguiente manera:

$$\mathfrak{C} = \{x \in \Omega : g(x) \leq 0, h(x) = 0\}$$

Definición 9. *(Restricciones activas). Consideremos el problema de optimización (2.2) y su conjunto factible $\mathfrak{C}$. Sea $x^* \in \mathfrak{C}$; entonces*

$$g_i(x) \leq 0 \text{ es activa en } x^* \Leftrightarrow g_i(x^*) = 0$$

en caso contrario la restricción es inactiva en x^*.

En el caso en que f sea cuadrática, g_i y h_i afines, cualquier mínimo local sera mínimo global, ya que el mínimo del problema es único por convexidad, y además siempre existirá por tratarse de un problema cuadrático.

Esta es una de las ventajas de las Máquinas de Vectores de Soporte frente a otros algoritmos de aprendizaje automático, y es que la solución siempre existe y es única.

Al tratarse de un problema de optimización con restricciones no podemos usar las condiciones suficientes y necesarias vistas anteriormente, para ello es necesario introducir la teoría Lagrangiana. Ésta teoría permite resolver problemas de optimización con restricciones.

2.1.3 Teoría Lagrangiana

El principal uso de la teoría Lagrangiana en este contexto es transformar un problema de optimización con restricciones de igualdad y desigualdad a un problema auxiliar para identificar el óptimo del problema original. Esta transformación será llevada a cabo a través de una nueva función que introduciremos conocida como función Lagrangiana.

Definimos la función Lagrangiana asociada al problema (2.2) como:

$$\mathscr{L}(x, \alpha, \beta) \quad = \quad f(x) + \sum_{i=1}^{k} \alpha_i g_i(x) + \sum_{i=1}^{m} \beta_i h_i(x) \tag{2.3}$$

donde α_i y β_j son los multiplicadores de Lagrange y se cumple que; f función convexa, g_i, h_i afines y $\alpha_i \geq 0$.

2.1.4 Formulación del Problema Dual

Dado el problema de optimización definido en (2.2), éste puede ser transformado a su correspondiente problema dual mediante la siguiente fórmula:

$$\begin{aligned} & \text{maximizar} \quad \theta(\alpha, \beta) = inf \mathscr{L}(x, \alpha, \beta) \\ \text{sujeto a} \quad & \alpha \geq 0, \end{aligned} \tag{2.4}$$

donde $\mathscr{L}$ es el Lagrangiano de la función (2.2) definida en (2.3).

Como podemos observar, en este nuevo problema tratamos de encontrar el máximo de una función θ, que sólo dependerá de los multiplicadores de Lagrange, i.e. α_i y β_j para todo i, j. De hecho, la función θ se define como el ínfimo del Lagrangiano sobre x.

En la mayoría de las ocasiones resolver el problema dual resulta mucho más sencillo y en el caso de las SVM, tiene una serie de ventajas que se explicarán más adelante. De la formulación dual, obtenemos varios teoremas que serán útiles en la búsqueda del óptimo.

Teorema 4. *(Teorema débil de la dualidad). Dado el problema de optimización* (2.2) *y su dual definido en* (2.4)*, si* $x \in \mathfrak{C}$ *es un punto de la región factible del problema primal y* (α, β) *es una solución factible del dual, existe la siguiente relación:*

$$f(x) \geq \theta(\alpha, \beta)$$

Esto se deduce inmediatamente de la definición (2.4) *del problema dual ya que:*

$$\theta(\alpha, \beta) = inf_{x \in \Omega} \mathscr{L}(x, \alpha, \beta) \leq \mathscr{L}(x, \alpha, \beta) = f(x) + \alpha g(x) + \beta h(x) \leq f(x)$$

Definición 10. *(Gap Dual). La diferencia entre el valor óptimo del problema primal* x^* *y de problema dual* (α^*, β^*) *es conocido como gap dual.*

Teorema 5. *(Teorema fuerte de la dualidad). Dado el problema de optimización* (2.2) *y su dual definido como* (2.4)*, si las restricciones son funciones afines, entonces el gap dual será cero. Es decir,*

$$f(x^*) = \theta(\alpha^*, \beta^*)$$

con $f(x^*)$ *y* $\theta(\alpha^*, \beta^*)$ *soluciones óptimas del problema primal y dual respectivamente.*

Las pruebas de estos teoremas se pueden encontrar en [4]. Resumiendo, el teorema débil de la dualidad da la relación entre la solución del problema dual y

primal, donde el valor de la función objetivo en el problema dual es menor o igual que el valor de la función objetivo en el problema primal. Por el teorema fuerte de la dualidad sabemos que el valor la función de optimización del problema primal y dual coinciden en el óptimo.

2.1.5 Condiciones Karush-Kuhn-Tucker

Las condiciones de Karush-Kuhn-Tucker (KKT) son el resultado analítico más importante en programación no lineal.

Para garantizar la existencia del mínimo del problema (2.2), se muestra el siguiente teorema, donde se especifican las condiciones de Karush-Kuhn-Tucker (KKT).

Teorema 6. *(Condiciones KKT). Una condición necesaria y suficiente para que un punto x^* sea óptimo del problema (2.2) es la existencia de (α^*, β^*) tal que:*

$$\frac{\partial \mathscr{L}(x^*, \alpha^*, \beta^*)}{\partial x_i} = 0 \quad i = 1, ..., n \tag{2.5}$$

$$\frac{\partial \mathscr{L}(x^*, \alpha^*, \beta^*)}{\partial \beta_i} = 0 \quad i = 1, ..., m \tag{2.6}$$

$$\alpha_i^* g_i(x^*) = 0 \quad i = 1, ..., k \tag{2.7}$$

$$g_i(x^*) \leq 0 \quad i = 1, ..., k \tag{2.8}$$

$$\alpha_i^* \geq 0 \quad i = 1, ..., k \tag{2.9}$$

La ecuación $\alpha_i^* g_i(x^*) = 0$, $i = 1, ..., k$ es conocida como la condición de complementariedad.

Una vez definidas la formulación primal y dual del problema de optimización, la función Lagrangiana y las condiciones KKT, se puede comenzar a estudiar la teoría básica de las Máquinas de Vectores de Soporte.

2.2 Aprendizaje Automatizado

El aprendizaje automatizado es una rama de la inteligencia artificial que contempla el desarrollo de algoritmos que son capaces de optimizar su performance usando datos de ejemplo o experiencia. El aprendizaje es necesario en casos donde no es posible escribir directamente un programa de computación para un problema dado. Estos casos se destacan por la falta de un conocimiento humano suficiente en el tema. Consideremos el reconocimiento del habla, es decir, la transformación de señales acústicas a texto. Esta tarea es llevada a cabo fácilmente por los humanos, aunque no somos capaces de explicar cómo. En aprendizaje automatizado, el enfoque dado es recolectar grandes cantidades de ejemplos de sonidos de diferentes personas y aprender a asociarlos con palabras.

2.3 Aprendizaje Supervisado

Existen varios tipos de problemas que se estudian en aprendizaje automatizado. Entre las fundamentales se encuentran el aprendizaje supervisado y el no supervisado. El aprendizaje no supervisado intenta descubrir como están organizados los datos provistos, agrupándolos por similaridad. Los datos usados por este tipo de algoritmos no se encuentran clasificados. En cambio, en el aprendizaje supervisado el objetivo es crear, a partir de un conjunto de datos de entrenamiento para los cuales se conoce la respuesta adecuada, un modelo numérico capaz de realizar predicciones precisas para datos nuevos. Los problemas de clasificación son una parte de los problemas supervizados en los que el objetivo es asociar con los datos un número discreto de valores, clases o categorías.

2.4 Clasificación

El problema de clasificación consiste formalmente en asignar una etiqueta o clase a un objeto. Cada objeto se describe por un conjunto de variables que representan medidas u observaciones sobre el mismo, y se le asocia un vector en un espacio

n-dimensional, donde cada dimensión representa una variable. Así, el objetivo es clasificar a un objeto $x \in \mathbb{R}^n$, donde a $\mathbb{R}^n$ se lo denomina "feature space" (espacio de variables o características). Cada objeto tiene asignada una clase ó etiqueta $l(x)$. Para un problema con c clases, $l(x)$ puede tomar c valores discretos distintos.

El objetivo entonces es encontrar una función clasificadora h tal que para cada objeto x sea $h(x) = l(x)$.

2.4.1 Error de Clasificación

Dado un conjunto de datos D, el error de clasificación del modelo h se define por

$$error(h) \quad = \quad \frac{N_{err}}{N_{tot}} \tag{2.10}$$

donde N_{err} es el número de clasificaciones erróneas de h en D y N_{tot} es el total de datos clasificados.

Formalmente, sea $h(x_i)$ la etiqueta o clase asignada por h a un objeto $x_i \in D$, y sea $l(x_i)$ la clase real correspondiente a ese objeto, entonces

$$error(h) \quad = \quad \frac{1}{N_{tot}} \sum_{i=1}^{N_{tot}} \{\omega(l(x_i), h(x_i))\} \tag{2.11}$$

donde $\omega(a, b) = 1$ si $a \neq b$ y $\omega(a, b) = 0$ si $a = b$.

2.4.2 Estrategias de Estimación del Error

Para encontrar una función clasificadora, un algoritmo necesita utilizar datos de donde poder aprender. Pero es conocido desde la estadística clásica que se necesita un nuevo conjunto de datos, independiente de los que se usaron para aprender, para poder estimar correctamente el error de un clasificador (estimación sin sesgo). Como en la práctica se dispone casi siempre de un conjunto limitado de datos, se utilizan

distintas estrategias para realizar eficazmente las dos tareas (aprender y estimar el error).

Dado un conjunto de datos D con N ejemplos, algunos de los métodos utilizados usualmente incluyen:

- **Resubstitution**: Se entrena el clasificador con D y se lo evalúa con D. El error tiene sesgo optimista y no es confiable.
- **Hold-Out**: Se particiona D en dos subconjuntos. Se utiliza una parte para entrenar el clasificador y la otra para estimar el error. Usualmente se repite el procedimiento varias veces para tener una estimación más confiable.
- **Bootstraps**: Se generan L subconjuntos de tamaño N eligiendo aleatoriamente y con repetición elementos de D. Se estima el error utilizando los ejemplos no incluidos en cada repetición.
- **K-fold Cross Validation**: Se particiona D en k subconjuntos de igual tamaño, y se utilizan $k-1$ subconjuntos para entrenar y el restante para evaluar. Esto se repite k veces, dejando para evaluar en cada repetición un subconjunto distinto.
- **Leave-One-Out**: Corresponde a utilizar K-fold Cross Validation con $K = N$. Es decir, en N repeticiones el clasificador es entrenado utilizando todos los datos excepto uno, al cual se lo utiliza para testear.

A continuación se presenta brevemente en el siguiente capítulo el tipo de aprendizaje supervisado que ha sido utilizado en este trabajo.

CAPÍTULO III
EXPLICACIÓN TEÓRICA DE LOS MODELOS Y ALGORITMOS

3.1 Máquinas de Vectores de Soporte

Las Máquinas de Vectores de Soporte o MVS son algoritmos (máquinas) que aprenden a discriminar entre miembros positivos y negativos de una clase de vectores de m-dimensional dada. Fue desarrollada en 1963 por Vapnik y Lerneren, pero se desarrolla y generaliza en los años noventa por Boser en 1992 y por Cortes y Vapnik en 1995. Fue ideada originalmente para la resolución de problemas de clasificación binarios en los que las clases eran linealmente separables [34]. Una MVS construye un hiperplano que puede ser utilizado para clasificar datos, por esta razón también se le conoce como Hiperplanos Óptimo de Separación o OSH ("Optimal Separating Hyperplane"), ya que la solución obtenida clasificaban de manera correcta todas las muestras en el espacio de entrada, colocando el hiperplano de separación lo más lejos posible de todas ellas [6]. Para lograr esto, se debe realizar un entrenamiento con los datos dispuestos para ese fin, a partir del cual se define el hiperplano óptimo, es decir, el plano que maximiza el margen de separación entre las muestras de cada clase.

El objetivo es que despúes de este entrenamiento, la máquina generalice bien para datos nuevos que no han participado en el entrenamiento, es decir, clasifique correctamente cada una de las muestras a la clase a la que pertenecen.

En este trabajo consideraremos la clasificación binaria, es decir, donde hay exactamente dos clases. Esta es la tarea más simple de clasificación. En este Capítulo son definidas las características teóricas de las MVS para problemas de clasificación con dos clases. Primero, definimos las Máquinas de Vectores de Soporte con para conjuntos de datos de entrenamiento linealmente separables en el espacio de entrada. Una vez concluido esto, nos extendemos al caso linealmente no separable y luego se generaliza para el caso no lineal donde es necesario trasladar el espacio de datos de entrada a un espacio de características de alta dimensión con el propósito de separar linealmente en el espacio de características.

3.1.1 Caso Linealmente Separable

Para solucionar un problema de clasificación binaria, la MVS debe establecer una superficie de decisión adecuada , basándose en el conjunto de datos de entrenamiento. La superficie de decisión es un hiperplano que separa los datos de entrenamiento en dos clases.

Se dispone entonces de un conjunto de n datos de entrenamientos, de la forma $(z_1, y_1), (z_2, y_2), \ldots, (z_n, y_n)$ donde $z \in \mathbb{R}^m$. Cada escalar y_i (llamado etiqueta) corresponderá a una de las dos clases que se identificarán como $+1$ y -1. Se llamará vector de etiquetas al vector $y = (y_1, y_2, \ldots, y_n)$.

Definición 11. *Un conjunto de vectores* $\{(z_1, y_1), (z_2, y_2), \ldots, (z_n, y_n)\}$ *donde* $z_i \in \mathbb{R}^m$ *e* $y_i \in \{-1, +1\}$ *para* $i = 1, ..., n$ *se dice linealmente separable si existe algún hiperplano en* $\mathbb{R}^m$ *que separa los vectores* $Z = \{z_1, z_2, \ldots, z_n\}$ *con etiquetas* $y_i = 1$ *de aquellos con etiqueta* $y_i = -1$.

En un problema linealmente separable existen muchos hiperplanos que pueden clasificar los datos. Pero las MVS no hallan uno cualquiera de estos hiperplanos, sino el único que maximiza la distancia entre él y el dato mas cercano de cada clase. Esta distancia es llamada margen, y el hiperplano que la maximiza se le llama Hiperplano

de Máximo Margen o Separación Óptima (HSO).

En la Figura 3-1 mostrada a continuación se puede observar un caso en $\mathbb{R}^2$ de datos linealmente con dos clases separadas por dos diferentes márgenes de separación. Como se ha dicho, el hiperplano óptimo será aquel que maximice el margen de separación entre las dos clases, y como se aprecia en este ejemplo, el margen 1 pareciera ser el que mejor separa las dos clases. También se puede observar que solo tres vectores (z_i) se encuentran justo a la distancia del margen 1 y 2.

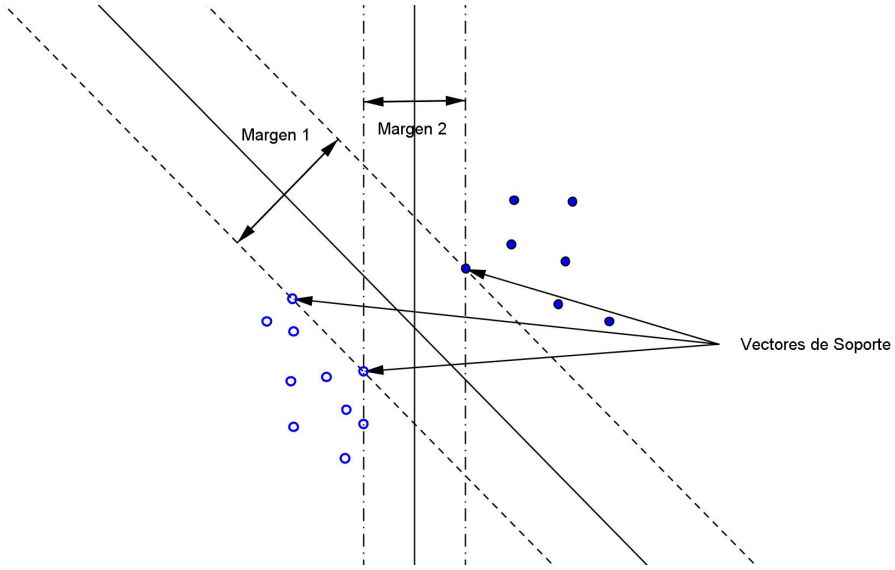

Figura 3–1: Maximización del margen

El hiperplano separador está dado de manera general por

$$H : z^t w + b \quad = \quad 0 \tag{3.1}$$

siendo b una constante que indica la posición del plano respecto al origen de coordenadas. Esta constante recibe el nombre de sesgo y define el umbral de decisión, w es el vector normal al hiperplano. El trabajo consiste en hallar el vector w que es normal al hiperplano, pero en el fondo es un vector de pesos que contiene la ponderación de cada dato, indicando qué tanto aportan en el proceso de clasificación.

La función discriminante (distancia) d será la función:

$$d(z,w,b) = \frac{|z^t w + b|}{\|w\|} \; con \; \|w\| = \sqrt{w^t w} \tag{3.2}$$

donde $\|w\|$ es la norma asociada al producto escalar en $\mathbb{R}^m$. Como se trata de datos linealmente separables, se pueden reescalar w y b, de tal manera que:

$$d(z,w,b) = \frac{1}{\|w\|} \Rightarrow |z^t w + b| = 1 \tag{3.3}$$

Los datos para los cuales $|z_i^t w + b| = 1$ se cumple tienen especial importancia. En particular, en los hiperplanos canónicos definidos por las ecuaciones:

$$z^t w + b = +1 \tag{3.4a}$$

$$z^t w + b = -1 \tag{3.4b}$$

se encuentran los datos z_i para los cuales una de estas restricciones (3.5) son activas:

$$z_i^t w + b \geq +1 \quad para \quad y_i = +1 \tag{3.5a}$$

$$z_i^t w + b \leq -1 \quad para \quad y_i = -1 \tag{3.5b}$$

Para datos linealmente separables estas restricciones clasifican de manera correcta a qué clase pertenece cada dato y se pueden combinar para expresarlo de una sola forma:

$$y_i(z_i^t w + b) \geq 1 \quad para \quad i = 1, .., n \tag{3.6}$$

Representando H_1 y H_2 los hiperplanos canónico tal como se aprecia en la Figura 3-2, la distancia perpendicular del origen al hiperplano H_1 ($z_i^t w + b = +1$) es $\frac{|1-b|}{\|w\|}$ y del origen al hiperplano H_2 ($z_i^t w + b = -1$) es $\frac{|-1-b|}{\|w\|}$. Aquellos objetos que se encuentran sobre los hiperplanos H_1 y H_2 son llamados Vectores de Soporte.

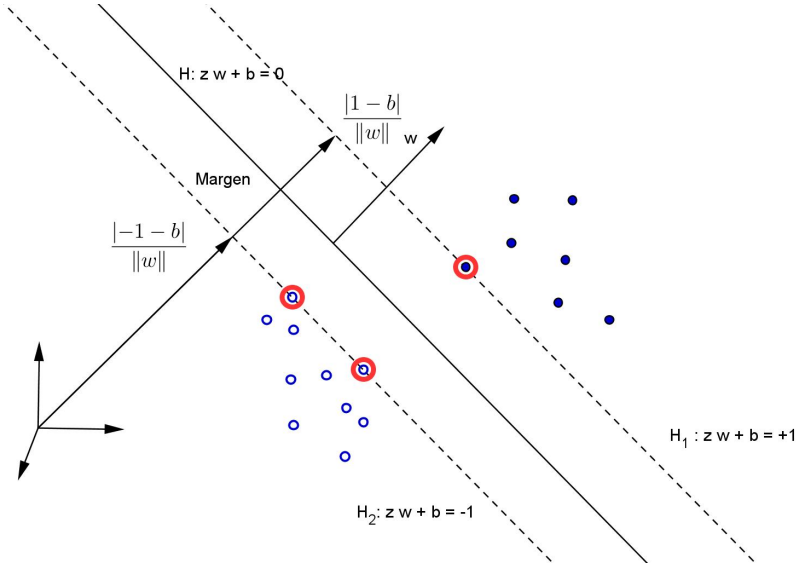

Figura 3–2: Caso linealmente separable

Eliminar cualquiera de los puntos que no se encuentran sobre H_1 y H_2 no afecta el resultado de la clasificación, pero eliminar alguno de los vectores de soporte si. La distancia entre los dos hiperplanos H_1 y H_2 es $\frac{2}{\|w\|}$, donde $\frac{1}{\|w\|}$ es la distancia entre el hiperplano definido por H respecto a cada uno de los hiperplanos canónicos. El objetivo para lograr una correcta clasificación es maximizar $\frac{2}{\|w\|}$ lo cual es equivalente a minimizar $\frac{\|w\|^2}{2}$. De esta manera, el problema de optimización a tratar es el siguiente:

$$\begin{aligned} (P) \underset{w,b}{\text{minimizar}} \quad & \frac{\|w\|^2}{2} \\ \text{sujeto a} \quad & y_i(z_i^t w + b) - 1 \geq 0 \quad para\ i = 1, ..., n \end{aligned} \tag{3.7}$$

donde n es el número de muestras de entrenamiento.

Este problema puede resolverse usando las técnicas de Programación Cuadrática ("Quadratic Programming" o QP por sus siglas en inglés). Esta solución se obtiene a través de las condiciones de optimalidad de primer orden, conocidas como las condiciones de Karush, Kuhn y Tucker (KKT) (2.5). Para esto pasaremos a una formulación Lagrangiana del problema (P) (2.1.4). Hay dos razones importantes

para hacer esto: la primera es que las restricciones de la ecuación (3.7) se sustituirán por restricciones sobre multiplicadores de Lagrange, que serán más fáciles de manejar, la segunda es que con esta reformulación del problema, los datos de entrenamiento solo aparecen en forma de productos escalares entre vectores. Esta propiedad es crucial para generalizar el procedimiento al caso no lineal como veremos mas adelante. Entonces para minimizar (3.7) se debe minimizar la función de Lagrange (3.8) sobre w y b, y luego maximizarla sobre los multiplicadores de Lagrange:

$$\mathscr{L}(w, b, x) = \frac{\|w\|^2}{2} - \sum_{i=1}^{n} x_i(y_i(z_i^t w + b) - 1) \tag{3.8}$$

donde los x_i son los multiplicadores de Lagrange no negativos asociados a las restricciones de (P). A partir de este Lagrangiano podemos plantear las condiciones KKT:

Condición del gradiente

$$\frac{\partial \mathscr{L}(\boldsymbol{w}, b, \boldsymbol{x})}{\partial \boldsymbol{w}} = w - \sum_{i=1}^{n} x_i y_i z_i = 0 \tag{3.9}$$

$$\frac{\partial \mathscr{L}(\boldsymbol{w}, b, \boldsymbol{x})}{\partial b} = \sum_{i=1}^{n} y_i x_i = 0 \tag{3.10}$$

$$\frac{\partial \mathscr{L}(\boldsymbol{w}, b, \boldsymbol{x})}{\partial x_i} = y_i(z_i^t w + b) - 1 = 0 \; para \; i = 1, .., n. \tag{3.11}$$

Condición de complementariedad

$$x_i(y_i(z_i^t w + b) - 1) = 0 \; para \; i = 1, .., n. \tag{3.12}$$

Condición de factibilidad primal

$$(y_i(z_i^t w + b) - 1) \geq 0 \; para \; i = 1, .., n. \tag{3.13}$$

Condición de no-negatividad

$$x_i \geq 0 \; para \; i = 1, .., n. \tag{3.14}$$

De las condiciones (3.9) y (3.10) se obtienen en el óptimo las siguientes relaciones:

$$w = \sum_{i=1}^{n} x_i y_i z_i \tag{3.15}$$

$$0 = \sum_{i=1}^{n} y_i x_i \tag{3.16}$$

Sustituyendo las ecuaciones (3.15) y (3.16) en la parte derecha de la función Lagrangiana se reduce la función a la forma dual con x_i como variable dual, en algunos ocaciones usaremos la función $\langle a, b\rangle = a^t b$ para representar con facilidad el producto escalar de dos vectores, los pasos de la sustitución de (3.15) se describen a continuación:

$$\begin{aligned}
\mathscr{L}(w,b,x) &= \frac{\|w\|^2}{2} - \sum_{i=1}^{n} x_i(y_i(z_i^t w + b) - 1) \\
&= \frac{1}{2}\langle w, w\rangle - \sum_{i=1}^{n} x_i(y_i(\langle w, z_i\rangle + b) - 1) \\
&= \frac{1}{2}\left\langle \sum_{i=1}^{n} x_i y_i z_i, \sum_{j=1}^{n} x_j y_j z_j \right\rangle - \sum_{i=1}^{n} x_i \left(y_i \left(\left\langle \sum_{j=1}^{n} x_j y_j z_j, z_i \right\rangle + b \right) - 1 \right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} x_i y_i x_j y_j \langle z_i, z_j\rangle - \sum_{i=1}^{n}\sum_{j=1}^{n} x_i y_i x_j y_j \langle z_i, z_j\rangle - b\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} x_i \\
&= -\frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} x_i y_i x_j y_j \langle z_i, z_j\rangle - b\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} x_i
\end{aligned}$$

Utilizando la ecuación (3.16) se llega a:

$$\mathscr{L}(w,b,x) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} x_i y_i x_j y_j \langle z_i, z_j\rangle + \sum_{i=1}^{n} x_i \tag{3.17}$$

De (3.17) el planteamiento del problema dual (2.1.4) es:

$$\begin{aligned} (D)\ \underset{x}{\text{maximizar}} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j y_i y_j \langle z_i, z_j\rangle + \sum_{i=1}^{n} x_i \\ \text{sujeto a} \quad & \sum_{i=1}^{n} x_i y_i = 0 \\ & x_i \geq 0,\ i = 1, \ldots, n. \end{aligned} \tag{3.18}$$

Teniendo en cuenta que $\max_x f(x)$ es equivalente a - $\min_x -f(x)$ y reformulando el (3.18) en forma matricial el nuevo problema de optimización es:

$$\begin{aligned} (D)\ \underset{x}{\text{minimizar}} \quad & \frac{1}{2}x^t Q x - e^t x \\ \text{sujeto a} \quad & y^t x = 0 \\ & x_i \geq 0,\ i = 1, ..., n \end{aligned} \tag{3.19}$$

donde $Q_{i,j} = y_i y_j \langle z_i, z_j\rangle$. Se puede notar que (D) es un problema de programación cuadrática. La resolución del problema (D) se reduce a la obtención de los valores óptimos para los multiplicadores x_i. Una vez conocidos estos valores, es posible obtener los valores óptimos de las variables primales del problema (P) representadas por el vector w mediante la ecuación (3.15).

Cabe destacar que la mayoría de los x_i son cero, ya que muy pocas restricciones de (3.7) son activas. Entonces un dato de entrenamiento z_i se denominará vector de soporte si el correspondiente $x_i > 0$. Por lo tanto el conjunto de indices de vectores de soporte ($x_i > 0$) se denotará como $N_{SV} = \{i | x_i > 0\}$.

De los desarrollos iniciales no se sigue una forma explícita de determinar el valor b, sin embargo, la condición KKT complementaria nos permite determinarlo. Para ello, basta elegir un $x_j > 0$ y despejar el valor de b obteniendo $b = y_j - z_j w$. Aunque se ha determinado b, es mas adecuado realizar los cálculos con todos los $x_j > 0$ y elegir como valor de b un valor promedio de los resultados obtenidos:

$$b \quad = \quad y_j - \sum_{i \in N_{SV}} x_i y_i \langle z_i, z_j\rangle \tag{3.20}$$

Una vez obtenidos el vector w y la constante b la clasificación de nuevos datos se realiza mediante la función:

$$h(z) \quad = \quad sgn\left(\sum_{i \in N_{SV}} y_i x_i \langle z, z_i \rangle + b\right) = sgn(H) \tag{3.21}$$

donde un nuevo dato es clasificado como $y_i = h(z_i) = +1$ si $H > 0$ y $y_i = h(z_i) = -1$ si $H < 0$.

3.1.2 Caso Linealmente no Separable

Todo el análisis desarrollado en la sección anterior se ha centrado en la búsqueda de un hiperplano de separación óptimo para conjuntos linealmente separables. Pero en la práctica no es habitual trabajar con conjuntos separables. Por lo general se encuentran datos de una clase dentro de la región correspondiente a los datos de otra clase y por tanto nunca podrán ser separados por medio de hiperplanos. En estas situaciones se diría que el conjunto es no separable. Para este caso, se le-vanta el supuesto de separabilidad lineal y se reformula el problema de optimización (P) expuesto para el caso linealmente separable. Estos cambios se traducen en la inclusión de variables de holguras. A continuación se detallan la forma en que se incorporan estas nuevas variables en la formulación del problema de optimización original (P).

La idea consistes en relajar o flexibilizar una o más restricciones del problema de optimización (P). Al relajar una restricción, se permite que uno o más objetos del conjunto de entrenamiento sean mal clasificados generando un error de clasificación no nulo. Esto genera un "trade-off" (compensación) para la obtención del OSH, ya que no solo se maximizará el margen de separación sino que también se debe de minimizar el error en la clasificación de los objetos que queden mal clasificados. Para incorporar estos dos objetivos se introduce un nuevo conjunto de variables de decisión a las que llamaremos variables de holgura. Estas variables modifican la formulación del problema de optimización (P), tanto en las restricciones como en la forma de la función objetivo. Las nuevas restricciones del problema quedan definidas

por:

$$z_i^t w + b \geq +1 - \xi_i \quad para \quad y_i = +1 \tag{3.22}$$

$$z_i^t w + b \leq -1 + \xi_i \quad para \quad y_i = -1 \tag{3.23}$$

$$\xi_i \geq 0 \; para \; i = 1, ..., n \tag{3.24}$$

Estas variables cuantifican el nivel de error del modelo para cada dato z_i. De la ecuación anterior es posible concluir que si un dato z_i está mal clasificado, el correspondiente valor de la variable ξ_i será mayor a 1, y para un dato bien clasificado el valor de ξ_i será igual a 0. También se puede considerar como un dato bien clasificado a aquel que cae dentro del margen ya que el error $\xi_i \in [0, 1]$.

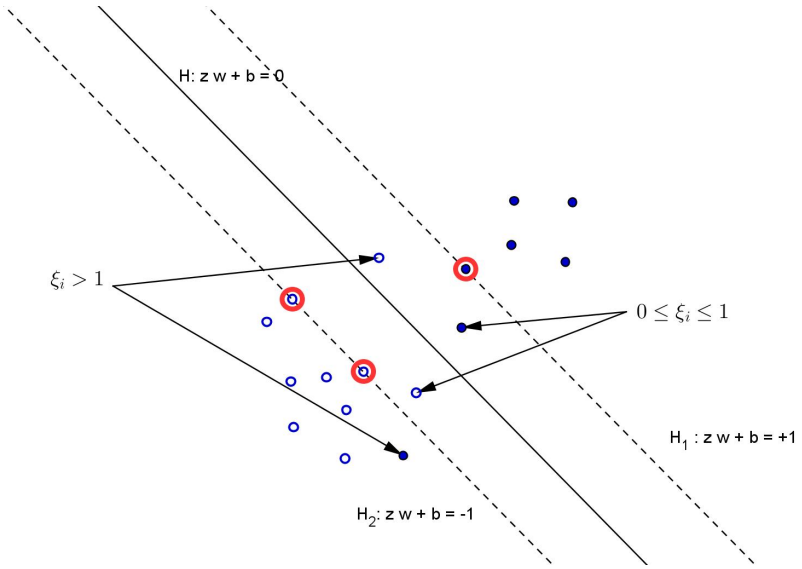

Figura 3–3: Caso linealmente no separable

Con este resultado es posible determinar una cota superior para el error correspondiente a la suma total de todas las variables ξ_i ($\sum_{i=1}^{n} \xi_i$). Una manera natural de incorporar este costo extra en la función objetivo es cambiándola por:

$$\underset{w,b,\xi}{\text{minimizar}} \quad \frac{\|w\|^2}{2} + C \sum_{i=1}^{n} \xi_i$$

Entonces la nueva formulación del problema queda definida por la expresión (3.25).

$$\begin{aligned} (P)\ \underset{w,b,\xi}{\text{minimizar}} \quad & \frac{\|w\|^2}{2} + C\sum_{i=1}^{n}\xi_i \\ \text{sujeto a} \quad & y_i(z_i^t w + b) - 1 + \xi_i \geq 0 \quad para\ i = 1, ..., n \\ & \xi_i \geq 0 \quad para\ i = 1, ..., n \end{aligned} \tag{3.25}$$

donde $\Xi = (\xi_1, ..., \xi_n)$ y C son parámetros determinados a priori los cuales pueden ser vistos como valores de penalización. Un valor pequeño de C maximiza el margen y el hiperplano es menos sensitivo a los datos anómalos en las muestras de entrenamiento; mientras que un valor grande de C minimiza el número de los puntos mal clasificados, esto quiere decir que $C > 0$ es la constante que controla el "trade-off".

La forma de enfrentar el problema (3.25) es similar a la utilizada para el caso linealmente separable. Primero se obtienen las condiciones de optimalidad (KKT) y luego se plantea el problema dual. La función Lagrangiana quedaría:

$$\mathscr{L}(w, b, x, \xi, \mu) = \frac{\|w\|^2}{2} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n} x_i(y_i(z_i^t w + b) - 1 + \xi_i) - \sum_{i=1}^{n}\mu_i\xi_i \tag{3.26}$$

donde μ_i son los multiplicadores de Lagrange introducidas por las restricciones $\xi_i \geq 0$. Las condiciones KKT para este problema seria:

Condición del gradiente

$$\frac{\partial\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\mu})}{\partial \boldsymbol{w}} = w - \sum_{i=1}^{n} x_i y_i z_i = 0 \tag{3.27}$$

$$\frac{\partial\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\mu})}{\partial b} = \sum_{i=1}^{n} y_i x_i = 0 \tag{3.28}$$

$$\frac{\partial\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\mu})}{\partial x_i} = y_i(z_i^t w + b) - 1 + \xi_i = 0\ para\ i = 1, .., n. \tag{3.29}$$

$$\frac{\partial\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\mu})}{\partial \xi_i} = C - x_i - \mu_i = 0\ para\ i = 1, .., n. \tag{3.30}$$

Condición de complementariedad

$$x_i(y_i(z_i^t w + b) - 1 + \xi_i) = 0 \; para \; i = 1, .., n. \tag{3.31}$$

$$\mu_i \xi_i = 0 \; para \; i = 1, .., n. \tag{3.32}$$

Condición de factibilidad primal

$$(y_i(z_i^t w + b) - 1 + \xi_i) \geq 0 \; para \; i = 1, .., n. \tag{3.33}$$

Condición de no-negatividad

$$\xi_i \geq 0 \; para \; i = 1, .., n. \tag{3.34}$$

$$x_i \geq 0 \; para \; i = 1, .., n. \tag{3.35}$$

$$\mu_i \geq 0 \; para \; i = 1, .., n. \tag{3.36}$$

Después se obtienen las relaciones (3.37) de las condiciones (3.27),(3.28) y (3.30):

$$w = \sum_{i=1}^{n} x_i y_i z_i \tag{3.37a}$$

$$0 = \sum_{i=1}^{n} y_i x_i \tag{3.37b}$$

$$C = x_i + \mu_i \tag{3.37c}$$

Nuevamente manipulando algebraicamente (3.26) se tiene:

$$\begin{aligned} \mathscr{L}(w, b, x, \xi, \mu) &= \frac{\|w\|^2}{2} + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} x_i(y_i(z_i^t w + b) - 1 + \xi_i) - \sum_{i=1}^{n} \mu_i \xi_i \\ &= \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} x_i y_i \langle w, z_i\rangle - b\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i \xi_i - \sum_{i=1}^{n} \mu_i \xi_i \\ &= \frac{1}{2}\langle w, w\rangle - \sum_{i=1}^{n} x_i y_i \langle w, z_i\rangle + \sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \xi_i(C - x_i - \mu_i) \end{aligned} \tag{3.38}$$

Utilizando (3.37a) y (3.37b) en (3.38)

$$\mathscr{L}(w, b, x, \xi, \mu) = \frac{1}{2}\langle w, w\rangle - \sum_{i=1}^{n} x_i y_i \langle w, z_i\rangle + \sum_{i=1}^{n} x_i \tag{3.39}$$

Reemplazando (3.37a) en (3.39)

$$\mathscr{L}(w,b,x,\xi,\mu) \;=\; -\frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} x_i y_i x_j y_j \langle z_i, z_j\rangle + \sum_{i=1}^{n} x_i \tag{3.40}$$

Por otra parte sabemos que $x_i, \mu_i \geq 0$ y al combinarlo con (3.37c) se obtiene:

$$0 \leq x_i \leq C \; para \; i = 1, ..., n \tag{3.41}$$

Finalmente el planteamiento del problema dual es:

$$\begin{aligned} (D)\ \underset{x}{\text{maximizar}} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j y_i y_j \langle z_i, z_j\rangle + \sum_{i=1}^{n} x_i \\ \text{sujeto a} \quad & \sum_{i=1}^{n} x_i y_i = 0 \\ & 0 \leq x_i \leq C,\ i = 1, \ldots, n. \end{aligned} \tag{3.42}$$

Reformulando el problema (3.42) en forma matricial y en términos de minimización, el nuevo problema de optimización es:

$$\begin{aligned} (D)\ \underset{x}{\text{minimizar}} \quad & \frac{1}{2}x^t Q x - e^t x \\ \text{sujeto a} \quad & y^t x = 0 \\ & 0 \leq x_i \leq C,\ i = 1, ..., n \end{aligned} \tag{3.43}$$

donde $Q_{i,j} = y_i y_j \langle z_i, z_j\rangle$.

El problema de optimización (3.43) es bastante similar a (3.19) con excepción de las restricciones sobre x_i, por otra parte la función solución al problema de clasificación $h(z)$ conserva la misma expresión de (3.21). Para calcular el umbral b usaremos las condiciones complementarias KKT. Cuando se observa la ecuación (3.32) y (3.30) se puede notar que si $\xi_i = 0$, entonces $x_i < C$ $(i \in N_{SV})$. Así que se puede simplificar el cálculo de b tomando los vectores de entrenamiento asociados a $0 < x_i < C$ y usar la ecuación (3.33) con $\xi_i = 0$, luego promediar este valor entre todos los vectores de soporte.

3.1.3 Caso no Lineal

En la mayoría de las aplicaciones del mundo real, el espacio de entradas Z por lo general, es un conjunto que no puede ser separado linealmente por un hiperplano gracias a la naturaleza de los datos, y debido a esta limitación, su tratamiento consiste en utilizar una función $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^f$, con m,$f \in \mathbb{N}$, $f \geq m$ ó $f = \infty$. Dicha función mapea la información de los datos de entrada a una de mayor dimensión llamado espacio de características y tiene como objetivo hacer que el conjunto obtenido $\phi(Z)$ sea un conjunto linealmente separable o en su caso poder minimizar el error mediante la separación con un hiperplano, es decir que el número de vectores clasificados incorrectamente sea mínimo.

$$Z = \{z_1, z_2, ..., z_n\} \rightarrow \phi(Z) = \{\phi(z_1), \phi(z_2), ..., \phi(z_n)\} \in \mathbb{R}^f$$

Pero como no es posible determinar a priori, dado el conjunto de entrada Z, una función ϕ que cumpla los objetivos descritos previamente, la representación por medio de funciones Kernel ofrece una solución al producto interno de las transformaciones de cada dato consideradas en su problema dual:

$$\begin{aligned} (D) \underset{x}{\text{maximizar}} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j y_i y_j \phi(z_i)^t \phi(z_j) + \sum_{i=1}^{n} x_i \\ \text{sujeto a} \quad & \sum_{i=1}^{n} x_i y_i = 0 \\ & 0 \leq x_i \leq C,\ i = 1, \ldots, n. \end{aligned} \tag{3.44}$$

A la función descrita en el capítulo I por $K(z_i, z_j) = \phi(z_i)^t \phi(z_i)$, se le conoce como función Kernel y su uso es más importante que el de la propia función ϕ, ya que con la función Kernel no se requiere su conocimiento de forma explícita.

En la Figura 3-4 se muestra un mapeo de un espacio de entradas de dos dimensiones a un espacio de características de tres dimensiones, donde la información no puede ser separada por una máquina lineal en el espacio de entrada mientras que en

el espacio de características esto resulta muy sencillo.

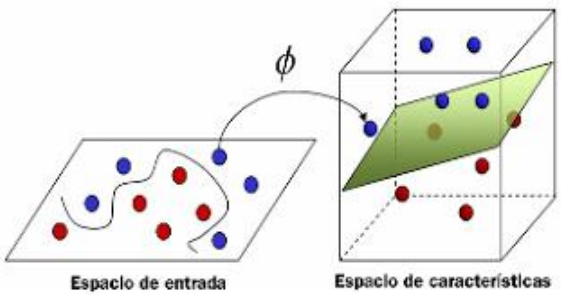

Figura 3–4: Mapeo del Espacio de entrada a un Espacio de características.

Existen distintas funciones Kernel que permiten adaptar la MVS a cada conjunto de muestras, con el fin de obtener mejores resultados. Algunos ejemplos de funciones Kernel [30] que han sido sugeridas o empleadas en problemas de clasificación son las siguientes:

- Kernel Lineal: $K(z_i, z_j) = z_i^t z_j$
- Kernel Polinomial: $K(z_i, z_j) = (r + \gamma z_i^t z_j)^d$, $\gamma > 0$, $r, d \in \mathbb{R}$
- Kernel Gaussiano: $K(z_i, z_j) = c \cdot exp(-\gamma \parallel z_i - z_j \parallel_2^2)$, $\gamma > 0$, $c \in \mathbb{R}$

Como se vio anteriormente en (3.44), una propiedad de las MVS en el caso lineal es que éstas pueden ser expresadas en una representación dual, esto significa que la ecuación (3.21) puede ser expresada como una combinación lineal de los puntos de entrenamiento. Por lo tanto, la regla de decisión puede ser evaluada usando productos punto

$$h(z) \quad = \quad sgn\left(\sum_{i \in N_{SV}} y_i x_i \phi(z)^t \phi(z_i) + b\right) \tag{3.45}$$

Pero gracias a los Kernels, (3.45) se convierte en:

$$h(z) \quad = \quad sgn\left(\sum_{i \in N_{SV}} y_i x_i K(z, z_i) + b\right) \tag{3.46}$$

Así mismo el problema dual de optimización para el caso no lineal que era dado por (3.42) se convierte en:

$$\begin{aligned} (D)\ \underset{x}{\text{maximizar}} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j y_i y_j K(z_i, z_j) + \sum_{i=1}^{n} x_i \\ \text{sujeto a} \quad & \sum_{i=1}^{n} x_i y_i = 0 \\ & 0 \leq x_i \leq C,\ i = 1, \ldots, n. \end{aligned} \tag{3.47}$$

Ahora (3.47) en términos de minimización y en forma matricial el problema de optimización cuadrática convexa que surge de las MVS es:

$$\begin{aligned} (D)\ \underset{x}{\text{minimizar}} \quad & \frac{1}{2}x^t Q x - e^t x \\ \text{sujeto a} \quad & y^t x = 0 \\ & 0 \leq x_i \leq C,\ i = 1, ..., n \end{aligned} \tag{3.48}$$

donde $Q_{i,j} = y_i y_j K(z_i, z_j)$.

Para hallar el umbral b, se hace de forma análoga a (3.20), pero ahora se promedia

$$b \quad = \quad y_j - \sum_{i \in N_{SV}} x_i y_i K(z_i, z_j) \tag{3.49}$$

para todos los puntos con $x_j > 0$.

3.2 Enfoques Previos para Entrenar una MVS

Entrenar una MVS es equivalente a resolver un problema de QP n-dimensional, donde n es el número de muestras en la matriz de datos de entrenamiento. Resolver este problema usando las técnicas tradicionales para QP involucra un número grande de operaciones, las cuales son muy costoso en tiempo y de recursos computacionales;

y en la mayoría de los casos esto no resulta práctico para problemas grandes. Producto a esta limitante se han creado varios algoritmos para solucionar el problema del entrenamiento de las MVS. Un pseudoalgoritmo para calcular MVS sería:

1. Seleccionar el parámetro C (que representa un compromiso entre minimizar el error de entrenamiento y la maximización del margen de separación entre los elementos de las clases), la función Kernel y cualquier otro parámetro requerido por la función Kernel empleada.
2. Resolver el problema de Programación Cuadrática o la formulación alternativa usando el algoritmo apropiado de Programación Cuadrática para obtener los vectores de soporte.
3. Obtener el umbral b usando los vectores de soporte ya calculados.
4. Obtener la función de desición.

Existen varios algoritmos para el entrenamiento de MVS. Por los resultados reportados se destacan:

3.2.1 SMO

El algoritmo de Optimización Mínima Secuencial ó como se le conoce "Secuencial Mínimal Optimización" [29], SMO por su siglas en inglés, es obtenido a partir de la idea del método de descomposición a su extremo, al optimizar un subconjunto mínimo de únicamente dos puntos en cada iteración (los algoritmos de descomposición son procedimientos iterativos que optimizan solamente un pequeño subconjunto de las variables en cada iteración y en el caso extremo se optimizan solo dos variables por iteración). El poder de esta técnica reside en el hecho de que el problema de optimización para dos puntos admite una solución analítica, eliminando la necesidad de usar un optimizador de programación cuadrática iterativo como parte del algoritmo.

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

Experimentalmente, el desempeño de SMO es muy bueno para MVS con entradas escasas, así como para MVS no lineales. Esto es debido a que el tiempo de computación del Kernel puede ser reducido, mejorando directamente su desempeño. A pesar de que necesita mas iteraciones para converger, cada iteración usa solo pocas operaciones, por lo tanto converge muy rápido. Además del tiempo de convergencia, otra característica del algoritmo radica en que este no necesita almacenar la matriz del Kernel en la memoria, ya que no se involucran operaciones matriciales. El algoritmo SMO se desempeña bien para problemas grandes, porque éste escala bien con el tamaño del conjunto de entrenamiento. Los autores de SMO aseguran que es un fuerte candidato para llegar a ser el algoritmo de entrenamiento estándar de MVS.

3.2.2 LIBSVM

LIBSVM es una librería para MVS creada en 2001 por Chih-Chung Chang y Chih-Jen Lin [7]. Se trata de un programa que soporta estimación de distribución, clasificación y regresión de Vectores de Soporte. La mayoría de los métodos de descomposición obtienen un conjunto de datos inicial a partir del conjunto de datos entero, este conjunto de datos es optimizado en cada iteración, mejorando el valor de la función objetivo en cada iteración. El proceso iterativo llega a su fin cuando un criterio de parada derivado de las condiciones de Karush-Kuhn-Tucker es satisfecho o una precisión requerida es alcanzada. El algoritmo LIBSVM está basado en el algoritmo SMO, sin embargo, posee una estrategia de descomposición llamada

selección de conjunto de trabajo [20] el cual es mucho mas avanzado. LIBSVM emplea un algoritmo de dirección de búsqueda que minimiza la función objetivo en cada iteración. El algoritmo inicia realizando una primera aproximación de la función objetivo obteniendo un vector x. A partir de esta primera aproximación calcula $x' = x + \lambda u$, donde la dirección u tiene únicamente dos coeficientes no cero. El algoritmo emplea dos direcciones de búsqueda, una dirección de búsqueda u^{ij} para λ positivos y una dirección de búsqueda $-u^{ij} = u^{ji}$ para λ negativo. La dirección de búsqueda mas efectiva para cada iteración será la dirección que minimiza la función objetivo.

3.2.3 ASL

Este es un algoritmo de punto interior, que se usa para resolver problemas de optimización con una sola restricción lineal de igualdad y las restricciones de caja. La dirección de búsqueda se obtiene mediante la aproximación del Hessiano de la función objetivo en el método de Newton por un múltiplo de la matriz de identidad. El algoritmo usa técnicas de descomposición y es eficiente para resolver problemas de optimización donde el Hessiano de la función objetivo es una matriz grande, densa, y posiblemente mal condicionado. Una implementación específica del algoritmo en el que se desarrolla la aproximación de Hessiano está dada por la fórmula cíclica Barzilai - Borwein (CBB) [9].

3.2.4 MPI

Entrenar específicamente una MVS es equivalente a resolver un problema de programación cuadrática con una restricción lineal y restricciones de caja donde el número de variables es igual al número de datos de entrenamiento. Este problema se hace bastante difícil cuando el tamaño de los datos es muy grande. Los algoritmos basados en los Métodos de Punto Interior (MPI), han permitido resolver problemas de programación cuadrática de gran escala, y son muy eficientes en aquellos problemas que son estrictamente convexos. Pero como no son apropiados para resolver problemas MVS, varios enfoques que utilizan la tecnología MPI con el objetivo de

resolver los problemas MVS han desarrollado en común el objetivo de explotar la estructura de bajo rango de la matriz Kernel [11], mientras que otros algoritmos se basan en el conocimiento de la estructura del subespacio nulo de la matriz Q para calcular de manera precisa la factorización de Cholesky [1].

En el capítulo siguiente se explicará como se deduce el algoritmo primal dual de punto interiores corrector-predictor (tradicional), junto con el algoritmo de bajo costo propuesto en esta investigación.

CAPÍTULO IV
ALGORITMOS DE PUNTOS INTERIORES PARA PROGRAMACIÓN CUADRÁTICA CONVEXA

4.1 Programación Cuadrática

Un problema de programación cuadrática (QP) es aquel que busca maximizar o minimizar una función objetivo cuadrática con respecto a un vector $x \in \mathbb{R}^n$ sujeto a restricciones lineales de igualdad o desigualdad. La programación cuadrática es muy usada ya que un gran número de problemas aparecen de forma natural como cuadráticos, pero además es importante porque aparece como un subproblema frecuentemente para resolver problemas de optimización no lineales más complicados. El problema cuadrático que se considera en esta investigación es la formulada en (4.1) descrito en los capítulo I y III ya que es el problema dual (D) del problema primal (P) (3.7) visto en el capítulo anterior.

$$\begin{aligned} \underset{x}{\text{minimizar}} \quad & \frac{1}{2}x^tQx - e^tx \\ \text{sujeto a} \quad & y^tx = b \\ & 0 \leq x \leq C \end{aligned} \tag{4.1}$$

4.2 Métodos Primal-Dual de Punto Interior

Los QP se pueden resolver usando diferentes métodos. En esta investigación se utilizará los métodos de punto interior que pertenecen a la clase de métodos de

programación lineal y no lineal. Para la resolución de programas cuadráticos los métodos de puntos interiores han demostrado que funciona bien en la práctica y en la teoría estos métodos están bien desarrollados. Por lo tanto los métodos de punto interior son ampliamente utilizados en aplicaciones.

Los métodos de punto interior resuelven los problemas iterativamente tales que en todas las iteraciones se satisfacen estrictamente las restricciones de no negatividad para algunas variables. Se acercan a la solución, ya sea del interior o exterior de la región factible, pero nunca se encuentran en el límite de este región. Cada iteración del método es computacionalmente costosa, pero se puede lograr un progreso significativo hacia la solución. Estas características particulares es lo que separa los métodos de punto interior de otros métodos.

Para establecer las ecuaciones que nos permitirán diseñar los métodos de punto interior que utiliza la teoría general de optimización con restricciones se hace mediante la definición de una función de Lagrange y junto con las condiciones Karush-Kuhn-Tucker (KKT) del QP que queremos resolver. Las condiciones de KKT, o las condiciones de optimalidad, son condiciones que deben ser satisfechas por un vector x para la solución de un QP dado. Los métodos de punto interior que se presentan en esta tesis resuelven el problema dual de forma simultánea con el problema primal y por esta razón estos métodos de punto interior se conocen también como métodos primal-dual de punto interior.

Fomulación del Procedimiento en General

En esta sección se aplicará las condiciones de optimalidad (KKT) vista en (2.5) al problema (4.1), también se mostrará cómo este método avanza en cada iteración hacia la solución.

Se establece primero la función Lagrangiana asociada al problema (4.1) en (4.2) donde s es el multiplicador de Lagrange asociada a la restricción lineal de igualdad, t y u multiplicadores asociados a las restricciones de caja.

$$\mathscr{L}(x,s,t,u) = \frac{1}{2}x^tQx - e^tx - s(y^tx - b) - t^tx - u^t(C-x) \tag{4.2}$$

El gradiente de la función Lagrangiana con respecto a x queda como sigue:

$$\nabla_x\mathscr{L}(x,s,t,u) = Qx - e - sy - t + u \tag{4.3}$$

Las condiciones necesarias y suficientes para resolver (4.2) son establecidas en (4.4), donde $i = 1,..,n$:

$$Qx - e - sy - t + u = 0 \tag{4.4a}$$
$$y^tx - b = 0 \tag{4.4b}$$
$$C - x \geq 0 \tag{4.4c}$$
$$t_ix_i = 0 \tag{4.4d}$$
$$u_i(C-x)_i = 0 \tag{4.4e}$$
$$(x,t,u) \geq 0 \tag{4.4f}$$

Introduciendo el vector de holgura $w_i = C - x_i$, con $w \geq 0$ se pueden reescribir estas condiciones como:

$$\begin{aligned} Qx - e - sy - t + u &= 0 && \text{(4.5a)} \\ y^t x - b &= 0 && \text{(4.5b)} \\ C - w - x &= 0 && \text{(4.5c)} \\ TXe &= 0 && \text{(4.5d)} \\ UWe &= 0 && \text{(4.5e)} \\ (x, t, u, w) &\geq 0 && \text{(4.5f)} \end{aligned}$$

donde $X = diag(x_1, ..., x_n)$, $T = diag(t_1, ..., t_n)$, $W = diag(w_1, ..., w_n)$, y $U = diag(u_1, ..., u_n)$.

Se define ahora una función $G(x, s, t, u, w)$ en (4.6) tal que las raíces de esta función sea la solución de las ecuaciones (4.5a)-(4.5e):

$$G(x, s, t, u, w) = \begin{bmatrix} r_e \\ r_b \\ r_c \\ r_{tx} \\ r_{uw} \end{bmatrix} = \begin{bmatrix} Qx - e - sy - t + u \\ y^t x - b \\ C - w - x \\ TXe \\ UWe \end{bmatrix} = 0 \qquad (4.6)$$

Aplicando el método de Newton a $G(x, s, t, u, w) = 0$ se obtiene una dirección de búsqueda en cada iteración, resolviendo el siguiente sistema de ecuaciones lineales:

$$J(x,s,t,u,w)\begin{bmatrix}\Delta x\\ \Delta s\\ \Delta t\\ \Delta u\\ \Delta w\end{bmatrix} = -F(x,s,t,u,w) \tag{4.7}$$

donde J es el Jacobiano de G. El método de Newton constituye un modelo lineal para G en torno al punto actual y obtiene la dirección de la búsqueda $(\Delta x, \Delta s, \Delta t, \Delta u, \Delta w)$ resolviendo el sistema (4.7). Para simplificar (4.7) se usan los vectores residuales r_e, r_b, r_c, r_{tx} y r_{uw} y escribiendo de forma explícita el Jacobiano se obtiene el siguiente sistema de ecuaciones:

$$\begin{bmatrix} Q & -y & -I & I & 0\\ y^t & 0 & 0 & 0 & 0\\ -I & 0 & 0 & 0 & -I\\ T & 0 & X & 0 & 0\\ 0 & 0 & 0 & W & U\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta s\\ \Delta t\\ \Delta u\\ \Delta w\end{bmatrix} = -\begin{bmatrix} r_e\\ r_b\\ r_c\\ r_{tx}\\ r_{uw}\end{bmatrix} \tag{4.8}$$

Al resolver este sistema de forma iterativa se debe llegar al punto óptimo. Sin embargo en la práctica este primer enfoque no es usado ya que un paso completo de Newton generalmente será infactible. Usar el método de Newton puede violar las restricciones de positividad $(x^+, t^+, u^+, w^+) > 0$ y para evitar esta dificultad se hace una búsqueda lineal a partir de la dirección de Newton, de manera que la nueva iteración queda:

$$(x^+, s^+, t^+, u^+, w^+) = (x, s, t, u, w) + \alpha(\Delta x, \Delta s, \Delta t, \Delta u, \Delta w)$$

donde α se escoge en el intervalo (0,1] de manera que se cumpla la desigualdad $(x^+, t^+, u^+, w^+) > 0$. Usualmente, α es muy cercano a cero, por lo que se hace muy poco progreso hacia la solución de (4.1) con la dirección obtenida en (4.8).

Aquí se puede notar que los métodos primal-dual de punto interior modifican el procedimiento básico de Newton en dos aspectos importantes:

1. Ellos desvían la dirección de búsqueda hacia el interior de las coordenadas no negativas $(x, t, u, w) \geq 0$ de manera que se pueda avanzar a través de la dirección antes que una de las componentes de (x, t, u, w) sea negativa.
2. Ellos evitan que las componentes (x, t, u, w) se muevan demasiado cerca de la frontera de las coordenadas no negativas. Las direcciones de búsqueda calculadas desde puntos muy cerca de la frontera tienden a distorsionar.

Si por ejemplo la ecuación $w_i u_i = 0$ se linealiza como se hace en Newton, se tiene que

$$w_i u_i + w_i \Delta u_i + u_i \Delta w_i = 0$$

entonces si una variable, digamos w_i, es cero, la ecuación de Newton queda $u_i \Delta w_i = 0$ con $u_i > 0$, por lo que el paso $\Delta w_i = 0$. Desde ese momento esa variable quedaría bloqueada en cero en todo el proceso, y como no se puede recuperar se tendría problemas para mejorar el camino al óptimo.

Por esta razón es necesario mantener todas las variables estrictamente positivas aunque sea poco. También es importante que todas las condiciones de complementariedad del tipo $w_i u_i$ y $t_i x_i$ converjan a cero al mismo ritmo:

$$\begin{aligned} w_i^k u_i^k &\approx \tau^k \to 0 \quad \textit{cuando} \quad k \to \infty \\ t_i^k x_i^k &\approx \tau^k \to 0 \quad \textit{cuando} \quad k \to \infty \end{aligned}$$

Con ese objetivo los algoritmos de punto interior, en algún momento, modifican las condiciones de complementaridad para centrar la trayectoria a seguir en el proceso.

4.2.1 Camino Central

El camino central $\mathscr{C}$ es un arco de puntos estrictamente positivos que está parametrizado por un escalar $\tau > 0$. Cada punto $(x_\tau, s_\tau, t_\tau, u_\tau, w_\tau)$ en $\mathscr{C}$ cumple estrictamente la desigualdad (4.5f) y además el sistema (4.6) con la diferencia que $TXe = \tau e$ y $UWe = \tau e$. De esta forma, se puede definir $\mathscr{C}$ como:

$$G(x_\tau, s_\tau, t_\tau, u_\tau, w_\tau) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \tau e \\ \tau e \end{bmatrix}, (x_\tau, t_\tau, u_\tau, w_\tau) > 0 \tag{4.9}$$

Se puede notar que los puntos sobre el camino central son estrictamente factible y que $r_e = 0$, $r_c = 0$ y $r_b = 0$ para estos puntos.

Si $\mathscr{C}$ converge a algún punto cuando τ tiende a cero, entonces debe converger a una solución primal-dual de (4.1). Lo que hacen los algoritmos primal-dual es tomar pasos de Newton hacia puntos en $\mathscr{C}$ para los cuales $\tau > 0$ en vez de tomar la dirección pura de Newton. Esto se explica porque los pasos de Newton hacia los puntos en $\mathscr{C}$ son sesgados hacia el interior del ortante definido por $(x, t, u, w) > 0$, y por consiguiente se pueden tomar pasos más grandes en esta dirección antes de violar las condiciones de positividad.

Esta dirección de búsqueda sesgada se describe de la siguiente forma: se introduce un parámetro de centrado $\sigma \in [0, 1]$ y una medida de dualidad μ:

$$\mu = \frac{x^t t + u^t w}{2n} \tag{4.10}$$

la cual da la medida del valor promedio de los productos $x_i t_i$ y $u_i w_i$. Si se reduce significativamente μ bastaría tomar un parámetro de centrado suficientemente pequeño, esto es hacer $\sigma = 0$ en este caso. Si en otro caso no se reduce μ, entonces

la dirección de búsqueda calculada no es tan útil y se elegiría un valor más grande para σ. Muchos algoritmos usan valores intermedios en el intervalo abierto (0,1) para σ con el fin de logra un equilibrio, en reducir μ y mantener una aproximación cercana al camino central.

Haciendo $\tau = \sigma\mu$ y aplicando nuevamente el método de Newton al sistema (4.9), la dirección $(\Delta x, \Delta s, \Delta t, \Delta u, \Delta w)$ es un paso de Newton hacia el punto $(x_{\sigma\mu}, s_{\sigma\mu}, t_{\sigma\mu}, u_{\sigma\mu}, w_{\sigma\mu}) \in \mathscr{C}$ (y no hacia el punto que satisface las condiciones de KKT), y se halla resolviendo:

$$\begin{bmatrix} Q & -y & -I & I & 0 \\ y^t & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & -I \\ T & 0 & X & 0 & 0 \\ 0 & 0 & 0 & W & U \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta t \\ \Delta u \\ \Delta w \end{bmatrix} = - \begin{bmatrix} 0 \\ 0 \\ 0 \\ r_{tx} - \tau e \\ r_{uw} - \tau e \end{bmatrix} \tag{4.11}$$

Con estos conceptos básicos, se puede definir la estructura general de los algoritmos prima-dual [35] para el problema cuadrático (4.1).

Algoritmo 1: Primal-Dual de Puntos Interiores.

Data: $s_0, (x_0, t_0, u_0, w_0) > 0, y, Q$

Inicio

para $k = 0, 1, 2...$ **hacer**

Resolver

$$\begin{bmatrix} Q & -y & -I & I & 0 \\ y^t & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & -I \\ T & 0 & X & 0 & 0 \\ 0 & 0 & 0 & W & U \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta s_k \\ \Delta t_k \\ \Delta u_k \\ \Delta w_k \end{bmatrix} = - \begin{bmatrix} 0 \\ 0 \\ 0 \\ r_{tx} - \tau e \\ r_{uw} - \tau e \end{bmatrix}$$

donde $\tau = \sigma\mu$, $\sigma \in [0,1]$, $\mu = \frac{x_k^t t_k + w_k^t u_k}{2n}$

Actualizar:

$$\begin{bmatrix} x_{k+1} \\ s_{k+1} \\ t_{k+1} \\ u_{k+1} \\ w_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ s_k \\ t_k \\ u_k \\ w_k \end{bmatrix} + \alpha_k \begin{bmatrix} \Delta x_k \\ \Delta s_k \\ \Delta t_k \\ \Delta u_k \\ \Delta w_k \end{bmatrix}$$

Seleccionamos α_k tal que $(x_{k+1}, t_{k+1}, u_{k+1}, w_{k+1}) > 0$

Hasta el momento se ha asumido que los puntos iniciales $(x_0, s_0, t_0, u_0, w_0)$ son estrictamente factibles y, en particular, que satisfacen las condiciones (4.5a)-(4.5c) y (4.5f) de manera estricta simultáneamente. No obstante, para la mayoría de los problemas, es difícil encontrar puntos iniciales factibles. Esta tarea puede ser siempre trivial si se reformula el sistema, pero la reformulación puede a veces introducir distorsiones que pueden complicar la resolución del mismo. Una alternativa es dada por los métodos de puntos interiores infactibles, los cuales requieren solamente que los puntos iniciales (x_0, t_0, u_0, w_0) sean positivos. La dirección de búsqueda necesita ser modificada de manera que el movimiento se realice tan cerca de la factibilidad como del centro. Para esto sólo se necesita un pequeño cambio a la ecuación (4.11) agregando los residuales r_e, r_b y r_c que se definen como:

$$r_e = Qx - e - sy - t + u, \qquad r_b = y^t x - b, \qquad r_c = C - w - x$$

Haciendo esta modificación en el lado derecho de la ecuación (4.11), la dirección de búsqueda continua siendo un paso de Newton con puntos $(x_\tau, s_\tau, t_\tau, u_\tau, w_\tau) \in \mathscr{C}$ y el sistema modificado quedaría como:

$$\begin{bmatrix} Q & -y & -I & I & 0 \\ y^t & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & -I \\ T & 0 & X & 0 & 0 \\ 0 & 0 & 0 & W & U \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta t \\ \Delta u \\ \Delta w \end{bmatrix} = - \begin{bmatrix} r_e \\ r_b \\ r_c \\ r_{tx} - \tau e \\ r_{uw} - \tau e \end{bmatrix} \tag{4.12}$$

La forma en que se seleccionan los parámetros σ_k y la longitud de paso α_k son cruciales para el comportamiento de los métodos. Diferentes técnicas para controlar estos parámetros, directa e indirectamente, dan lugar a una variedad de métodos con diferentes propiedades teóricas.

4.2.2 Longitud de Paso

En esta sección se muestra como tomar una longitud de paso suficientemente buena de manera que las direcciones de búsqueda no violen las restricciones de positividad $(x_k, t_k, u_k, w_k) > 0$.

Varios métodos se pueden aplicar para calcular la longitud de paso, sus diferentes enfoques pueden influir en la rapidez con que el algoritmo converge. Para empezar se muestra la ecuación (4.13), donde se tiene un parámetro $\alpha \in [0, 1]$ (longitud de paso) para cinco direcciones $(\Delta x, \Delta s, \Delta t, \Delta u, \Delta w)$:

$$\begin{bmatrix} x_{k+1} \\ s_{k+1} \\ t_{k+1} \\ u_{k+1} \\ w_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ s_k \\ t_k \\ u_k \\ w_k \end{bmatrix} + \alpha \begin{bmatrix} \Delta x_k \\ \Delta s_k \\ \Delta t_k \\ \Delta u_k \\ \Delta w_k \end{bmatrix} \tag{4.13}$$

Concretamente lo que se quiere es elegir el $\hat{\alpha}$ mas grande tales que (4.14) sean satisfechas.

$$x_k + \hat{\alpha}\Delta x_k \geq 0 \tag{4.14a}$$

$$t_k + \hat{\alpha}\Delta t_k \geq 0 \tag{4.14b}$$

$$u_k + \hat{\alpha}\Delta u_k \geq 0 \tag{4.14c}$$

$$w_k + \hat{\alpha}\Delta w_k \geq 0 \tag{4.14d}$$

Al mirar estas ecuaciones por separado, se propone que $\hat{\alpha_x}$ sea la longitud de paso para la ecuación (4.14a), como también los son $\hat{\alpha_t}$,$\hat{\alpha_u}$,$\hat{\alpha_w}$ para las ecuaciones (4.14b),(4.14c),(4.14d) respectivamente, de modo que $\hat{\alpha} = \min\{\hat{\alpha_x}, \hat{\alpha_t}, \hat{\alpha_u}, \hat{\alpha_w}\}$.

Para cada una de las cuatro ecuaciones se consideran tres casos al tomar en cuenta el signo de Δx, Δt, Δu y Δw. Los tres casos para la ecuación (4.14a) se muestra en (4.15):

$$\Delta x > 0 : \hat{\alpha_x} = 1 \tag{4.15a}$$

$$\Delta x = 0 : \hat{\alpha_x} = 1 \tag{4.15b}$$

$$\Delta x_i < 0 : x_i + \hat{\alpha_x}\Delta x_i = 0 \quad \Rightarrow \quad \hat{\alpha_x} = \frac{-x_i}{\Delta x_i} \tag{4.15c}$$

Si cada uno de los elementos del vector Δx son positivos o igual a cero (4.14a) se satisface para cualquier $\hat{\alpha_x} \in [0,1]$, porque estamos tomando puntos $x > 0$. En

este caso la maxima longitud de paso que se pude tomar es haciendo $\alpha_x = 1$ porque es la longitud de paso usada en el método de Newton. En el caso de que algunos elementos de Δx sean negativos entonces se necesita tomar un $\hat{\alpha_x}$ tal que su valor sea el más pequeño de (4.15c).

De los casos anteriores se puede ver que $\hat{\alpha_x}$ puede ser tomado como se indica en

$$\hat{\alpha_x} = \min_{i:\Delta x_i<0} \left(1, \min \frac{-x_i}{\Delta x_i}\right) \tag{4.16}$$

Los mismo argumentos pueden ser usados para los tres casos mencionados en las ecuaciones (4.14b), (4.14c) y (4.14d) donde $\hat{\alpha_t}$, $\hat{\alpha_u}$ y $\hat{\alpha_w}$ se eligen de manera equivalente a $\hat{\alpha_x}$ como se muestran en:

$$\hat{\alpha_t} = \min_{i:\Delta t_i<0} \left(1, \min \frac{-t_i}{\Delta t_i}\right) \tag{4.17}$$

$$\hat{\alpha_u} = \min_{i:\Delta u_i<0} \left(1, \min \frac{-u_i}{\Delta u_i}\right) \tag{4.18}$$

$$\hat{\alpha_w} = \min_{i:\Delta w_i<0} \left(1, \min \frac{-w_i}{\Delta w_i}\right) \tag{4.19}$$

Las longitudes de paso de las variables primales y duales no pueden ser diferentes como en el caso de programación lineal (Q=0) ya que ellas están relacionas con la matriz Hessiana, y el usar diferentes longitudes de paso para las variables primales y duales puede desviar la factibilidad de la ecuación (4.5a). Por esta razón se elige la longitud de paso $\alpha = \eta\hat{\alpha}$, con $\hat{\alpha} = \min\{\hat{\alpha_x}, \hat{\alpha_t}, \hat{\alpha_u}, \hat{\alpha_w}\}$ donde el valor $\eta \in [0.9,1)$ se escoge tal que $\eta \to 1$ cerca de la solución, para de esta forma acelerar la convergencia asintótica. Se puede escoger $\eta = 0.99$ según [35].

4.2.3 Método Predictor-Correcto Mehrotra

El método primal dual de punto interior presentado en la sección (4.2) ya no es el método mas usado comúnmente en la práctica. Una variante aparece poco después y es conocida como el método predictor-corrector Mehrotra [26], convirtiéndose en unos de los métodos mas populares. La principal diferencia entre el enfoque de

predictor-corrector y el algoritmo anterior es la elección de la dirección de búsqueda.

Las características principales del algoritmo predictor-corrector de Mehrotra [26] son dos: (1) se añade un paso corrector a la dirección de búsqueda para que el algoritmo siga una trayectoria más cercana al conjunto de soluciones de (4.1) y (2) el parámetro de centrado se escoge luego de calcular la dirección de búsqueda. En cada iteración, se calcula el predictor (que es la dirección de búsqueda pura de Newton) y se evalúa cuán útil resulta como dirección de búsqueda. Si esta dirección provoca que μ se reduzca sin violar las condiciones de positividad $(x_k, t_k, u_k, w_k) > 0$, se necesita poco centrado y por consiguiente, se toma cercano a cero, para luego calcular una dirección de búsqueda centrada con este valor de σ. Si, por el contrario, la dirección pura de Newton no es productiva, entonces se escoge el parámetro de centrado cercano a 1.

Primero se comienza resolviendo el sistema (4.20) obteniendo la dirección de escalado afín $(\Delta x_a, \Delta s_a, \Delta t_a, \Delta u_a, \Delta w_a)$ y se determina la longitud de paso α_a tal como se describe en (4.2.2) de manera que se cumpla la factibilidad (4.14). El cálculo de este dirección se conoce también como el paso predictor (en este caso $\sigma = 0$ y la dirección es de descenso puro de Newton).

$$\begin{bmatrix} Q & -y & -I & I & 0 \\ y^t & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & -I \\ T & 0 & X & 0 & 0 \\ 0 & 0 & 0 & W & U \end{bmatrix} \begin{bmatrix} \Delta x_a \\ \Delta s_a \\ \Delta t_a \\ \Delta u_a \\ \Delta w_a \end{bmatrix} = - \begin{bmatrix} r_e \\ r_b \\ r_c \\ r_{tx} \\ r_{uw} \end{bmatrix} \tag{4.20}$$

Se define μ_a como el valor de μ que se obtendría con un paso completo hacia la frontera:

$$\mu^a = \frac{(x+\alpha_a\Delta x_a)^t(t+\alpha_a\Delta t_a)+(u+\alpha_a\Delta u_a)^t(w+\alpha_a\Delta w_a)}{2n} \tag{4.21}$$

Luego se calcula el parámetro de centrado σ usando la formula:

$$\sigma = \left(\frac{\mu_a}{\mu}\right)^3 \tag{4.22}$$

Este valor tiene la siguiente propiedad: cuando se hace un buen progreso en la dirección del predictor, $\mu_a \ll \mu$, y por ende, σ es un valor pequeño (se necesita poco centrado) y viceversa.

Entonces la dirección de búsqueda (Δx,Δs,Δt,Δu,Δw) se obtiene resolviendo un segundo sistema lineal que incorpora información de segundo orden obtenida del paso anterior y el parámetro centrado. Este sistema se muestra en (4.23) y se le llama paso corrector y centrado.

$$\begin{bmatrix} Q & -y & -I & I & 0 \\ y^t & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & -I \\ T & 0 & X & 0 & 0 \\ 0 & 0 & 0 & W & U \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta t \\ \Delta u \\ \Delta w \end{bmatrix} = - \begin{bmatrix} r_e \\ r_b \\ r_c \\ r_{tx} + \Delta T_a \Delta X_a - \tau e \\ r_{uw} + \Delta U_a \Delta W_a - \tau e \end{bmatrix} \tag{4.23}$$

donde $\Delta T_a = diag(\Delta t_a^1, \ldots, \Delta t_a^n)$, $\Delta X_a = diag(\Delta x_a^1, \ldots, \Delta x_a^n)$, $\Delta U_a = diag(\Delta u_a^1, \ldots, \Delta u_a^n)$ y $\Delta W_a = diag(\Delta w_a^1, \ldots, \Delta w_a^n)$. Esta variante reduce significativamente el número de iteraciones del método primal dual de punto interior.

Por último se calcula la longitud de paso global de la iteración, α, que cumpla la factibilidad de (4.14), mediante un último parámetro de seguridad o amortiguación, η, al nuevo punto

$$\begin{bmatrix} x_{k+1} \\ s_{k+1} \\ t_{k+1} \\ u_{k+1} \\ w_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ s_k \\ t_k \\ u_k \\ w_k \end{bmatrix} + \alpha \begin{bmatrix} \Delta x_k \\ \Delta s_k \\ \Delta t_k \\ \Delta u_k \\ \Delta w_k \end{bmatrix} \tag{4.24}$$

donde $\alpha = \min\{1, \eta\hat{\alpha}\}$ y $\eta \in [0.9,1.0)$.

4.2.4 Método Tradicional

El método predictor-corrector Mehrotra se indica a continuación. Nótese que el algoritmo requiere un punto de partida inicial $(x_0, s_0, t_0, u_0, w_0)$. El punto de partida no necesita estar en la región factible. Solo se necesita exigir que $(x_0, t_0, u_0, w_0) > 0$ de modo que los residuales sean cercanos a cero.

Algoritmo 2: Algoritmo Predictor-Corrector Mehrotra

Parámetros;

$\eta = 0.95$, $\sigma \in (0, 1)$;

Inicializar MPI;

iteración $k = 0$;

punto primal-dual $(x_0, t_0, u_0, w_0) > 0$ y $s_0 = 0$;

medida de dualidad $\mu = \frac{x_0^t t_0 + u_0^t w_0}{2n}$;

infactibilidad dual y primal $r_e^0, r_b^0, r_c^0, r_{tx}^0$ y r_{uw}^0;

Data: y,Q,C

Inicio

Mientras *condiciones de parada no satisfechas* **hacer**

Calcular el paso predictor $(\Delta x_a, \Delta s_a, \Delta t_a, \Delta u_a, \Delta w_a)$ (4.20)

Calcular la longitud de paso α_a

$$\alpha_a = \left[\max_{z=1,..,n}\{1, -\Delta x_z/x_z, -\Delta w_z/w_z, -\Delta t_z/t_z, -\Delta u_z/u_z\}\right]^{-1} \quad (4.25)$$

Calcular μ_a (4.21)

Calcular σ (4.22)

Calcular el paso corrector $(\Delta x, \Delta s, \Delta t, \Delta u, \Delta w)$ (4.23)

Calcular la longitud de paso $\alpha_{primal}, \alpha_{dual} > 0$ tal que

$(x_{k+1}, t_{k+1}, u_{k+1}, w_{k+1}) > 0$

$$\alpha_{primal} = \eta\left[\max_{z=1,..,n}\{1, -\Delta x_z/x_z, -\Delta w_z/w_z\}\right]^{-1} \quad (4.26)$$

$$\alpha_{dual} = \eta\left[\max_{z=1,..,n}\{1, -\Delta t_z/t_z, -\Delta u_z/u_z\}\right]^{-1} \quad (4.27)$$

Actualizar $(x_{k+1}, s_{k+1}, t_{k+1}, u_{k+1}, w_{k+1})$

$$(x_{k+1}, w_{k+1}) = (x_k, w_k) + \alpha_{primal}(\Delta x_k, \Delta w_k) \quad (4.28)$$

$$(s_{k+1}, t_{k+1}, u_{k+1}) = (s_k, t_k, u_k) + \alpha_{dual}(\Delta s_k, \Delta t_k, \Delta u_k) \quad (4.29)$$

Actualizar los residuales $r_e^{k+1}, r_b^{k+1}, r_c^{k+1}, r_{tx}^{k+1}, r_{uw}^{k+1}$ y la medida de dualidad μ

Hacer $k = k + 1$

4.2.5 Reducción del Sistema Lineal

Los sistemas lineales constituyen uno de los aspectos fundamentales en el comportamiento de los algoritmos de puntos interiores. Generalmente, estos sistemas son mal condicionados cerca del conjunto de soluciones del problema e influencian el comportamiento numérico de los algoritmos.

Como se puede observar en el método tradicional (4.2.4), el mayor esfuerzo computacional realizado en los métodos primal-dual se encuentra al resolver sistemas lineales como (4.20) y (4.23) para obtener la dirección $(\Delta x_k, \Delta s_k, \Delta t_k, \Delta u_k, \Delta w_k)$. Por esto, se transforma el sistema a resolver en un sistema equivalente de menor dimensión y con alguna estructura que favorezca su resolución. La diferencia entre algunos métodos de puntos interiores radica en la manera como los sistemas se plantean y como se resuelven.

Se puede reescribir el original sistema de ecuaciones lineales (4.8) de la siguiente manera:

$$Q\Delta x - y\Delta s - \Delta t + \Delta u = -r_e \tag{4.30}$$

$$y^t\Delta x = -r_b \tag{4.31}$$

$$-\Delta x - \Delta w = -r_c \tag{4.32}$$

$$T\Delta x + X\Delta t = -r_{tx} \tag{4.33}$$

$$W\Delta u + U\Delta w = -r_{uw} \tag{4.34}$$

Debido a que el punto (x, s, t, u, w) tiene componentes x, t, u y w estrictamente positivas, las matrices diagonales X, T, U y W son no singulares. Por lo tanto, multiplicando la ecuación (4.33) por X^{-1} y la ecuación (4.34) por W^{-1} se observa que:

$$\Delta t = X^{-1}\left(-r_{tx} - T\Delta x\right) \tag{4.35}$$

$$\Delta u = W^{-1}\left(-r_{uw} - U\Delta w\right) \tag{4.36}$$

Sustituyendo (4.35) y (4.36) en (4.30) se obtiene:

$$Q\Delta x - y\Delta s + X^{-1}r_{tx} + X^{-1}T\Delta x - W^{-1}r_{uw} - W^{-1}U\Delta x = -r_e \quad (4.37)$$

Despejando de (4.32) a Δw y sustituyendo en (4.37) se tiene:

$$\left(Q + X^{-1}T + W^{-1}U\right)\Delta x - y\Delta s = -r_e - X^{-1}r_{tx} + W^{-1}r_{uw} + W^{-1}Ur_c \quad (4.38)$$

Se puede escribir entonces el siguiente sistema equivalente del sistema (4.8):

$$\begin{bmatrix} Q + X^{-1}T + W^{-1}U & -y \\ y^t & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_e - X^{-1}r_{tx} + W^{-1}\left(r_{uw} + Ur_c\right) \\ -r_b \end{bmatrix} \quad (4.39)$$

$$\Delta t = X^{-1}\left(-r_{tx} - T\Delta x\right) \quad (4.40)$$

$$\Delta w = r_c - \Delta x \quad (4.41)$$

$$\Delta u = W^{-1}\left(-r_{uw} - U\Delta w\right) \quad (4.42)$$

El propósito de utilizar este nuevo sistema es que tiene ventajas en la estabilidad y rapidez. En el método tradicional se resuelve el sistema (4.39).

4.3 Una Propuesta de Bajo Costo

Este trabajo consiste en proponer un algoritmo primal-dual de punto interior de bajo costo para resolver el problema de optimización (4.1). Esta alternativa está basada en la idea del método ASL [13]. Se trata de sustituir la matriz Hessiana Q por la matriz de la identidad $\lambda_k I$ en (4.39) donde λ_k se toma como: $\max\{\lambda_0, \lambda_k^{BB}\}$ [3]. En consecuencia, el algoritmo de punto interior primal-dual modificado que se propone es el siguiente:

$$\begin{bmatrix} \lambda_k I + X^{-1}T + W^{-1}U & -y \\ y^t & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_e - X^{-1}r_{tx} + W^{-1}\left(r_{uw} + Ur_c\right) \\ -r_b \end{bmatrix} \quad (4.43)$$

Modificamos el sistema (4.39) sustituyendo Q por $\lambda_k I$. Al hacer esta sustitución se puede lograr despejar las direcciones de búsqueda ya que $\lambda_k I + X^{-1}T + W^{-1}U$ es una matriz diagonal. Así ya no es necesario resolver en cada iteración un sistema de ecuaciones lineales sino que ahora las variables se actualizarán en cada iteración.

Sea $D = \lambda_k I + X^{-1}T + W^{-1}U$, $b_1 = -r_e - X^{-1}r_{tx} + W^{-1}\left(r_{uw} + Ur_c\right)$ y $b_2 = -r_b$, de manera que el sistema (4.43) simplificado será:

$$\begin{bmatrix} D & -y \\ y^t & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{4.44}$$

Donde las ecuaciones lineales de (4.44) se muestran como:

$$D\Delta x - y\Delta s = b_1 \tag{4.45}$$

$$y^t \Delta x = b_2 \tag{4.46}$$

Multiplicando (4.45) por D^{-1} se obtiene la dirección $\Delta x = D^{-1}(b_1 + y\Delta s)$. La dirección Δs se consigue multiplicando (4.45) por $y^t D^{-1}$ así:

$$y^t \Delta x - y^t D^{-1} y \Delta s = y^t D^{-1} b_1 \tag{4.47}$$

$$b_2 - y^t D^{-1} y \Delta s = y^t D^{-1} b_1 \tag{4.48}$$

$$\Delta s = \frac{y^t D^{-1} b_1 - b_2}{-y^t D^{-1} y} \tag{4.49}$$

o sustituyendo Δx en (4.46). De modo que las direcciones de búsqueda quedan establecidas de la siguiente manera:

$$\Delta s = \frac{y^t D^{-1} b_1 - b_2}{-y^t D^{-1} y} \tag{4.50a}$$

$$\Delta x = D^{-1}(b_1 + y\Delta s_a) \tag{4.50b}$$

$$\Delta t = X^{-1}\left(-r_{tx} - T\Delta x_a\right) \tag{4.50c}$$

$$\Delta w = r_c - \Delta x_a \tag{4.50d}$$

$$\Delta u = W^{-1}\left(-r_{uw} - U\Delta w_a\right) \tag{4.50e}$$

La longitud espectral λ_k, como se había mencionado en el capítulo I, es la solución del siguiente problema de optimización:

$$\min_{\lambda \geq \lambda_0} \|\lambda(x_k - x_{k-1}) - (\nabla f(x_k) - \nabla f(x_{k-1}))\|_2 \tag{4.51}$$

Donde hacemos $s_k = x_k - x_{k-1}$, $y_k = g_k - g_{k-1}$ y $g_k = g(x_k) = \nabla f(x_k)$.

Por la definición de $||.||_2$ y recordando que minimizar $||w||_2$ es equivalente a minimizar $w^t w$ para algún vector $w \in \mathbb{R}^n$ se tiene lo siguiente:

$$\Phi(\lambda) = ||\lambda s_k - y_k||_2^2 \tag{4.52}$$

$$= (\lambda s_k - y_k)^t(\lambda s_k - y_k) \tag{4.53}$$

$$= (\lambda s_k^t - y_k^t)(\lambda s_k - y_k) \tag{4.54}$$

$$= \lambda^2 s_k^t s_k - 2\lambda s_k^t y_k + y_k^t y_k \tag{4.55}$$

Para dar solución al problema (4.51) se aplican las condiciones de optimalidad para problemas sin restricciones:

$$\nabla_\lambda \Phi(\lambda) = \lambda s_k^t s_k - 2 s_k^t y_k \Leftrightarrow \lambda_k^{BB} = \frac{s_k^t y_k}{s_k^t s_k} \tag{4.56}$$

$$\nabla_{\lambda\lambda} \Phi(\lambda) = 2 s_k^t s_k = 2||s_k||_2^2 \geq 0 \tag{4.57}$$

4.3.1 Algoritmo de Bajo Costo Predictor-Corrector

Algoritmo 3: Algoritmo de Bajo Costo Predictor-Corrector

Parámetros;

$\eta = 0.95$, $\sigma \in (0,1)$, $\lambda_0 > 0$;

Inicializar MPI;

iteración $k = 0$;

punto primal-dual $(x_0, t_0, u_0, w_0) > 0$ y $s_0 = 0$;

medida de dualidad $\mu = \frac{x_0^t t_0 + u_0^t w_0}{2n}$;

infactibilidad dual y primal $r_e^0, r_b^0, r_c^0, r_{tx}^0$ y r_{uw}^0;

Data: y,Q,C

Inicio

Mientras *condiciones de parada no satisfechas* **hacer**

Calcular el paso predictor $(\Delta x_a, \Delta s_a, \Delta t_a, \Delta u_a, \Delta w_a)$ (4.50)

donde $D = \lambda_k I + X^{-1}T + W^{-1}U$,

$b_1 = -r_e - X^{-1} r_{tx} + W^{-1}(r_{uw} + U r_c)$, $b_2 = -r_b$, $r_{tx} = TXe$ y

$r_{uw} = UWe$

Calcular la longitud de paso α_a

$$\alpha_a = \left[\max_{z=1,..,n} \{1, -\Delta x_z / x_z, -\Delta w_z / w_z, -\Delta t_z / t_z, -\Delta u_z / u_z\} \right]^{-1}$$

Calcular μ_a (4.21)

Calcular σ (4.22)

Calcular el paso corrector $(\Delta x, \Delta s, \Delta t, \Delta u, \Delta w)$ (4.50)

donde $r_{tx} = r_{tx} + \Delta T_a \Delta X_a - \sigma \mu e$ y $r_{uw} = r_{uw} + \Delta U_a \Delta W_a - \sigma \mu e$

Calcular la longitud de paso $\alpha_{primal}, \alpha_{dual} > 0$ tal que

$(x_{k+1}, t_{k+1}, u_{k+1}, w_{k+1}) > 0$

$$\alpha_{primal} = \eta \left[\max_{z=1,..,n} \{1, -\Delta x_z / x_z, -\Delta w_z / w_z\} \right]^{-1}$$

$$\alpha_{dual} = \eta \left[\max_{z=1,..,n} \{1, -\Delta t_z / t_z, -\Delta u_z / u_z\} \right]^{-1}$$

Actualizar $(x_{k+1}, s_{k+1}, t_{k+1}, u_{k+1}, w_{k+1})$

$$(x_{k+1}, w_{k+1}) = (x_k, w_k) + \alpha_{primal}(\Delta x_k, \Delta w_k)$$

$$(s_{k+1}, t_{k+1}, u_{k+1}) = (s_k, t_k, u_k) + \alpha_{dual}(\Delta s_k, \Delta t_k, \Delta u_k)$$

Actualizar los residuales $r_e^{k+1}, r_b^{k+1}, r_c^{k+1}, r_{tx}^{k+1}, r_{uw}^{k+1}$ y la medida de dualidad μ

Calcular la longitud espectral $\lambda_k = \max\left\{\lambda_0, \lambda_k^{BB}\right\}$ (4.57)

Hacer $k = k + 1$

4.4 Modificaciones del Método Propuesto

En esta sección se modifica el algoritmo de bajo costo predictor-corrector ya que al comenzar estar cerca de la solución el paso corrector hace que se pierda factibilidad dual y en algunas veces la convergencia del método.

4.4.1 Algoritmo de Bajo Costo Predictor-Corrector Modificado

La primera modificación que se la hace al método es la elección del paso corrector-centrado.

Algoritmo 4: Algoritmo de Bajo Costo Predictor-Corrector Modificado

Parámetros;

$\eta = 0.95$, $\sigma \in (0,1)$ $\lambda_0 > 0$;

Inicializar MPI;

iteración $k = 0$;

punto primal-dual $(x_0, t_0, u_0, w_0) > 0$ y $s_0 = 0$;

medida de dualidad $\mu = \frac{x_0^t t_0 + u_0^t w_0}{2n}$;

infactibilidad dual y primal $r_e^0, r_b^0, r_c^0, r_{tx}^0$ y r_{uw}^0;

Data: y,Q,C

Inicio

  Mientras *condiciones de parada no satisfechas* **hacer**

    Calcular el paso predictor $(\Delta x_a, \Delta s_a, \Delta t_a, \Delta u_a, \Delta w_a)$ (4.50)

    donde $D = \lambda_k I + X^{-1}T + W^{-1}U$,

    $b_1 = -r_e - X^{-1}r_{tx} + W^{-1}(r_{uw} + Ur_c)$, $b_2 = -r_b$, $r_{tx} = TXe$ y

    $r_{uw} = UWe$

    Calcular la longitud de paso α_a como en (4.25)

    Calcular μ_a (4.21)

    Calcular σ (4.22)

    Si $||r_c|| > 10^{-5}$ *y* $||r_b|| > 10^{-5}$ **entonces**

      Calcular el paso completo corrector y centrado resolviendo (4.50)

      con $r_{tx} = TXe - \sigma\mu + \Delta X \Delta Te$ y $r_{uw} = UWe - \sigma\mu + \Delta U \Delta We$

    sino

      Calcular el paso de centrado resolviendo (4.50) con

      $r_{tx} = TXe - \sigma\mu$ y $r_{uw} = UWe - \sigma\mu$

    Calcular la longitud de paso $\alpha_{primal}, \alpha_{dual} > 0$ tal que

    $(x_{k+1}, t_{k+1}, u_{k+1}, w_{k+1}) > 0$ como en (4.26) y (4.27)

    Actualizar $(x_{k+1}, s_{k+1}, t_{k+1}, u_{k+1}, w_{k+1})$ como en (4.28) y (4.29)

    Actualizar los residuales $r_e^{k+1}, r_b^{k+1}, r_c^{k+1}, r_{tx}^{k+1}, r_{uw}^{k+1}$ y la medida de

    dualidad μ

    Calcular la longitud espectral $\lambda_k = \max\left\{\lambda_0, \lambda_k^{BB}\right\}$ (4.57)

    Hacer $k = k + 1$

4.4.2 Algoritmo de Bajo Costo sin Paso Predictor

En esta otra modificación, el algoritmo de bajo costo no realiza el paso predictor.

Algoritmo 5: Algoritmo de Bajo Costo sin Paso Predictor

Parámetros;

$\eta = 0.95$, $\sigma = 0.5$, $\lambda_0 > 0$;

Inicializar MPI;

iteración $k = 0$;

punto primal-dual $(x_0, t_0, u_0, w_0) > 0$ y $s_0 = 0$;

medida de dualidad $\mu = \frac{x_0^t t_0 + u_0^t w_0}{2n}$;

infactibilidad dual y primal $r_e^0, r_b^0, r_c^0, r_{tx}^0$ y r_{uw}^0;

Data: y,Q,C

Inicio

Mientras *condiciones de parada no satisfechas* **hacer**

Calcular el paso de centrado resolviendo (4.50) con $r_{tx} = TXe - \sigma\mu$ y $r_{uw} = UWe - \sigma\mu$

Se determina la longitud de paso $\alpha_{primal}, \alpha_{dual} > 0$ tal que $(x_{k+1}, t_{k+1}, u_{k+1}, w_{k+1}) > 0$ como en (4.26) y (4.27)

Actualizar $(x_{k+1}, s_{k+1}, t_{k+1}, u_{k+1}, w_{k+1})$

$$(x_{k+1}, w_{k+1}) = (x_k, w_k) + \alpha_{primal}(\Delta x_k, \Delta w_k)$$

$$(s_{k+1}, t_{k+1}, u_{k+1}) = (s_k, t_k, u_k) + \alpha_{dual}(\Delta s_k, \Delta t_k, \Delta u_k)$$

Actualizar los residuales $r_e^{k+1}, r_b^{k+1}, r_c^{k+1}, r_{tx}^{k+1}, r_{uw}^{k+1}$ y la medida de dualidad μ

Calcular la longitud espectral $\lambda_k = \max\left\{\lambda_0, \lambda_k^{BB}\right\}$ (4.57)

Hacer $k = k + 1$

4.4.3 Algoritmo de Bajo Costo con Identificación de Variables

En esta sección se presentan estrategias para mejorar los métodos de bajo costo previamente diseñados.

Selección de Variables

La selección de variables es el proceso mediante el cual se selecciona un subconjunto de las variables originales de un problema. Este proceso reduce considerablemente la cantidad de variables utilizadas produciendo varios beneficios: facilidad para visualizar y entender los datos, eliminación de variables irrelevantes o redundantes, reducción de los requerimientos de medición y almacenamiento, reducción de los tiempos de entrenamiento.

Un proceso típico de selección de variables es un loop que consiste en cuatro pasos básicos: selección de un subconjunto de variables, evaluación del mismo, aplicación de un criterio de terminación y validación del resultado final.

En este trabajo hemos utilizado un tipo de selección de variables implementado en [19], para seleccionar las variables que están más cerca de los planos canónicos. Luego de seleccionar las variables candidatas en el algoritmo de bajo costo con identificación de variable (si hace la selección), se resuelve un subproblema más pequeño con el algoritmo de bajo costo sin el paso predictor. A continuación se derán los detalles de este método.

Selección del Subconjunto

En esta sección se muestra como la idea de identificación de variables puede ser aplicada al problema de MVS usando los valores más pequeños de la diagonal

$$\Theta = X^{-1}T + W^{-1}U$$

Los elementos de la diagonal Θ son $t_i/x_i + u_i/w_i$. Por complementariedad, cuando μ se aproxima a cero, sucede que si $t_i \to 0$ entonces $x_i \to x_i^* > 0$ o si $x_i \to 0$

entonces $t_i \to t_i^* > 0$ (análogamente, si $u_i \to 0$ entonces $x_i \to x_i^* < C$ o si $x_i \to C$ entonces $u_i \to u_i^* > 0$). Por otra parte, siempre que una variable converge a cero, esta converge con la misma rapidez que el parámetro μ. Así, siempre que $t_i \to 0$ y $u_i \to 0$, entonces Θ_i converge a cero con la misma rapidez que μ. De otra forma, Θ_i converge a infinito con la misma rapidez que $1/\mu$. Ejemplo, si $\mu = 10^{-8}$, entonces algunos elementos de la diagonal de Θ son de orden 10^{-8} mientras que otros son de orden 10^8.

Por esta razón se sabe que las variables $0 < x_i^* < C$ asociados a grandes valores de Θ_i^{-1} son aquellos datos z_i que están más cerca a los hiperplanos canónicos y son llamados vectores de soporte no acotados, cuando $x_i = C$ se le llaman vectores de soporte acotado. Sea

$$\Theta_i^{-1} = \frac{w_i x_i}{x_i u_i + w_i t_i} \tag{4.58}$$

El valor Θ_i^{-1} se hace muy grande cuando u_i y t_i tienden a cero y por las condiciones de complementariedad se cumple que $w_i + x_i = C$, es decir que $x_i \in (0, C)$. Si en la solución óptima cualquiera de ellos se hace cero, por las condiciones de complementariedad, cualquiera x_i o w_i es cero, y así Θ_i^{-1} es cero en la solución óptima como se observa en la Tabla 4–1:

Tabla 4–1: Clasificación de los vectores y no vectores de soporte. Aquí α_i^* es la variable de holgura asociada a la i-ésima restricción de (3.25) y definida como $\alpha_i^* = y_i(z_i \cdot w + b) - 1 + \xi_i$.

Tipo de Dato	x_i^*	α_i^*	ξ_i^*
Vector de Soporte acotado	C	0	$(0, \infty)$
Vector de Soporte no acotado	$(0, C)$	0	0
No vector de soporte	0	$(0, \infty)$	0

No es posible identificar los vectores de soporte sino hasta que se logre encontrar el máximo margen de separación entre las dos clases. Sin embargo en cada iteración podemos obtener valores de Θ_i, eligiendo entre ellos prospectos de vectores

de soporte con valores pequeños de Θ_i.

Como la identificación intermedia en cada iteración se aproxima cada vez al máximo margen de separación, es claro que estos patrones están más propensos a ser vectores de soporte. Esto nos permite reducir el tamaño del conjunto de índices. Para medir cómo la clasificación intermedia se acerca al óptimo, nosotros podemos usar la medida de dualidad μ la cual converge a cero. En práctica hemos establecido q como el tamaño de nuestro conjunto de índices cuyo valor estará entre q_L y q_U:

$$q = \max(q_L, \min(\lceil \rho n \rceil, q_U))$$

donde n es el número de datos de entrenamiento y $\rho = \mu^{\frac{1}{\beta}}$ se usa para disminuir el tamaño del conjunto de índices con la medida de dualidad. Aquí $\beta > 0$ es un parámetro para controlar la velocidad de decrecimiento cuando μ converge a cero.

En nuestro trabajo escogemos en cada iteración se escoge $q_U = |\{i : x_{i+1}/x_i > 0.9\}|$ como cota superior del número de vectores de soporte, esto basado en los indicadores de Tapia [2]. Como el crecimiento y decremento de Θ_i^{-1} depende de la divergencia y convergencia de $1/\mu$ y μ, se pueden separar estos dos diferentes tipos de Θ_i^{-1} usando $\sqrt{\mu}$. Definimos $Q(z, q)$ como un subconjunto del conjunto de $M = \{1, .., n\}$ que contiene los índices de los q componentes más pequeños de z.

$$\mathbf{Q}(z, q) = \{Q | Q \subseteq M, |Q| = q \quad y \quad z_i \leq z_j \quad \forall i \in Q, j \notin Q\}$$

Luego tenemos que $Q(\Theta, q)$ es un subconjunto de M que contiene los q índices más pequeños de Θ_i. Definimos una cota inferior sobre el número de índices de tamaño q_L constituidos por los valores grandes de Θ_i^{-1}

$$q_L \quad = \quad |\{i, \Theta_i^{-1} \geq \theta\sqrt{\mu}\}| \tag{4.59}$$

donde θ es un parámetro prescrito. Cuando hay convergencia del método, el parámetro q_L convergerá eventualmente al número de valores en que diverge Θ_i^{-1}, o equivalentemente, al número de vectores de soporte.

Si el número de etiquetas de una clase es mayor que la otra clase, esta elección debido al desbalance que existe entre los patrones elegidos de las clases $+1$ y -1, podría no elegir patrones de una clase en el subconjunto $Q(\Theta, q)$, y es importante que el subconjunto contenga vectores de soporte con índices en ambas clases. Para evitar esta situción desfavorable, lo que se quiere es utilizar una variedad equilibrada de patrones, que especifique el número de q^+ y q^- elegidos en cada clase como:

$$\begin{aligned} q^+ &= \max\left(q_L^+, \min\left(\left\lceil \frac{\min(\lceil \rho n \rceil, q_u)}{2} \right\rceil, n^+\right)\right) \\ q^- &= \max\left(q_L^-, \min\left(\left\lceil \frac{\min(\lceil \rho n \rceil, q_u)}{2} \right\rceil, n^-\right)\right) \end{aligned}$$

donde n^+ y n^- son los números de $+1$ y -1 patrones, respectivamente. Luego de ajustar q^+ y q^- tenemos $q^+ + q^- = q$. Las cotas inferiores, q_L^+ y q_L^-, son determinadas similarmente a (4.59) para cada clase como:

$$\begin{aligned} q_L^+ &= |\{i, \Theta_i^{-1} \geq \theta\sqrt{\mu} \quad y \quad y_i = +1\}| \\ q_L^- &= |\{i, \Theta_i^{-1} \geq \theta\sqrt{\mu} \quad y \quad y_i = -1\}| \end{aligned}$$

Ahora definimos los subconjuntos de las q^+/q^- componentes más pequeñas para cada clases como:

$$\begin{aligned} Q^+(\Theta, q^+) &= \{Q | Q \subseteq M, |Q| = q^+ \quad y \quad z_i \leq z_j \quad \forall i \in Q, j \notin Q \quad y \quad y_i = y_j = +1\} \\ Q^-(\Theta, q^-) &= \{Q | Q \subseteq M, |Q| = q^- \quad y \quad z_i \leq z_j \quad \forall i \in Q, j \notin Q \quad y \quad y_i = y_j = -1\} \end{aligned}$$

En nuestra propuesta no se verifica en cada iteración los subconjuntos $\mathbf{Q}^-(z, q^-)$ y $Q^+(z, q^+)$ sino que se hace la selección de un subconjunto de datos de entrenamiento candidatos cuando el número de cifras significativas es mayor que 1. Al cumplir esta condición y seleccionadas las variables, inmediatamente resolvemos este subconjunto como un nuevo problema donde se ignora la gran mayoría de datos mal

clasificados. Esta idea nos brinda una manera de descargar memoria para el algoritmo de bajo costo, pero no se obtiene el margen máximo de separación sino solo una aproximación.

Algoritmo 6: Algoritmo de Bajo Costo con Identificación de Variable

Parámetros;

$\eta = 0.95$, $\sigma \in (0,1)$, $\lambda_0 > 0$, $\beta = 4$, $\theta = 100$;

Inicializar MPI;

iteración $k = 0$;

punto primal-dual $(x_0, t_0, u_0, w_0) > 0$ y $s_0 = 0$;

medida de dualidad $\mu = \frac{x_0^t t_0 + u_0^t w_0}{2n}$;

infactibilidad dual y primal $r_e^0, r_b^0, r_c^0, r_{tx}^0$ y r_{uw}^0;

Data: y,Q,C

Inicio

Mientras *condiciones de parada no satisfechas* **hacer**

Calcular el paso predictor $(\Delta x_a, \Delta s_a, \Delta t_a, \Delta u_a, \Delta w_a)$ (4.50)

donde $D = \lambda_k I + X^{-1}T + W^{-1}U$,

$b_1 = -r_e - X^{-1}r_{tx} + W^{-1}(r_{uw} + Ur_c)$, $b_2 = -r_b$, $r_{tx} = TXe$ y

$r_{uw} = UWe$

Calcular la longitud de paso α_a como en (4.25)

Calcular μ_a (4.21)

Calcular σ (4.22)

Si $||r_c|| > 10^{-5}$ *y* $||r_b|| > 10^{-5}$ **entonces**

Calcular el paso completo corrector y centrado resolviendo (4.50)

con $r_{tx} = TXe - \sigma\mu + \Delta X \Delta Te$ y $r_{uw} = UWe - \sigma\mu + \Delta U \Delta We$

sino

Calcular el paso de centrado resolviendo (4.50) con

$r_{tx} = TXe - \sigma\mu$ y $r_{uw} = UWe - \sigma\mu$

Calcular la longitud de paso α_{primal}, α_{dual} como en (4.26) y (4.27)

Actualizar $(x_{k+1}, s_{k+1}, t_{k+1}, u_{k+1}, w_{k+1})$ como en (4.28) y (4.29)

Actualizar los residuales $r_e^{k+1}, r_b^{k+1}, r_c^{k+1}, r_{tx}^{k+1}, r_{uw}^{k+1}$ y la medida de dualidad μ

Calcular la longitud espectral $\lambda_k = \max\left\{\lambda_0, \lambda_k^{BB}\right\}$ (4.57)

Calcular $sigfig = \max\left(-\lg_{10} \frac{|primal_{obj} - dual_{obj}|}{|primal_{obj}|+1}, 0\right)$

Si *sigfig* > 1 **entonces**

Calcular q^+ y q^-

Seleccionar el conjunto de indices con: $Q(\Theta^{-1}, q^+, q^-)$

Salir

Hacer $k = k + 1$

4.4.4 Punto Inicial

Los métodos de puntos interiores infactibles convergen para cualquier punto inicial (puntos iniciales positivos) pero las experiencia en la práctica han mostrado que el punto inicial debería estar centrado (no en las cotas) y los residuales iniciales deberían ser pequeños, en otro caso se requerirá un mayor número de iteraciones.

En esta investigación se propone el punto inicial $(x_0, s_0, t_0, u_0, w_0)$ como:
Si $C \geq 1$ entonces $x_0 = e \cdot C, w_0 = x_0, t_0 = 1/Ce, u_0 = t_0$ y $s_0 = 0$
Sino $x_0 = e \cdot C, w_0 = x_0, t_0 = x_0, u_0 = x_0$ y $s_0 = 0$, donde e es un vector de unos.

4.4.5 Criterio de Parada

Para detener el proceso iterativo, y por lo tanto el método, es necesario definir algunos criterios de parada para detectar cuándo el método ha alcanzado la solución.

El algoritmo de punto interior trabaja simultáneamente en los espacios primal y dual. Se alcanza la optimalidad cuando las variables primales y duales logran ser factibles y cuando las condiciones de complementariedad se cumplen. Por lo tanto es de interés observar las siguientes cuatro medidas donde $\epsilon = 10^{-5}$:

- Medida relativa de la infactibilidad primal:

$$\frac{\|y^t x - b\|}{\|b\| + 1} \geq \epsilon \tag{4.60}$$

$$\frac{\|C - x - w\|}{\|C\| + 1} \geq \epsilon \tag{4.61}$$

- Medida relativa de la infactibilidad dual:

$$\frac{\|Qx - e - sy - t + u\|}{\|e\| + 1} \geq \epsilon \tag{4.62}$$

- Complementaridad absoluta:

$$\left| \frac{x^t t + u^t w}{2n} \right| \quad \geq \quad \epsilon \tag{4.63}$$

Para explicar la última condición de parada es necesario definir el problema dual asociado a (4.1):

$$\begin{aligned} \underset{u,s}{\text{maximizar}} \quad & -\frac{1}{2} x^T Q x + bs - u^t C e \\ \text{sujeto a} \quad & Hx - e - y^T x + u = t \\ & u, t, s \geq 0 \end{aligned} \tag{4.64}$$

Las reglas de parada por defecto es parar cuando se cumple las condiciones (4.60)-(4.63), pero permitiremos que el algoritmo se detenga solo cuando sus funciones objetivo acuerdan seis cifras significativas.

Las cifras significativa entre el valor de la función objetivo primal y dual se calcula de la siguiente manera:

$$sigfig = \max\left(-\lg_{10} \frac{\mid primal_{obj} - dual_{obj} \mid}{\mid primal_{obj} \mid +1}, 0 \right)$$

Es posible utilizar criterios de parada más avanzados tales que se encuentran en algunos casos especiales necesarios para que el algoritmo converja.

En esta presentación los criterios de parada y la selección del punto inicial son suficientes y se aplican a todos los algoritmos de punto interior descritos anteriormente.

CAPÍTULO V
PROPUESTA: DISEÑO DEL SOFTWARE

MATLAB es un entorno de computación orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos. MATLAB es el nombre abreviado de "MATrix LABoratory". Se trata de un software matemático de alto nivel para realizar cálculos numéricos con vectores y matrices. Para ciertas operaciones es muy rápido, cuando puede ejecutar sus funciones en código nativo con los tamaños más adecuados para aprovechar sus capacidades de vectorización. En otras aplicaciones resulta bastante más lento que el código equivalente desarrollado en otros lenguajes como C/C++ o Fortran. Sin embargo, siempre es una magnífica herramienta para desarrollar aplicaciones técnicas, fácil de utilizar (a diferencia de los lenguajes de bajo nivel) y que aumenta significativamente la productividad de los programadores respecto a otros entornos de desarrollo. De este modo, MATLAB [25] es el lenguaje idóneo para programar un entrenamiento de pocas muestras de una MVS: sencillo, intuitivo, capaz de llevar a cabo numerosas operaciones matemáticas, rápido de programar y que trabaja muy bien con vectores y matrices.

5.1 Funciones MEX

Los ficheros MEX permiten llamar desde MATLAB a funciones escritas en C/C++ como si fueran funciones propias de MATLAB. Al compilar estas funciones se generan librerías compartidas, las denominadas funciones MEX. Estas funciones son ejecutables de extensión $*.dll$, $*.mexw32$ o $*.mexglx$ que pueden ser cargadas

y ejecutadas por MATLAB de forma automática.

Las funciones MEX tienen varias aplicaciones:

- Evitan tener que reescribir en MATLAB funciones que ya han sido escritas en C.
- Por motivos de eficiencia puede ser interesante reescribir en C las funciones críticas que consumen mas CPU del programa.

Para poder trabajar con los ficheros MEX, primero hay que seleccionar el compilador que se va a utilizar. Esto se hace mediante la siguiente instrucción en el prompt de MATLAB:

```
>> mex -setup
```

El compilador utilizado en el presente proyecto ha sido GCC de Ubuntu. La sintaxis del comando para compilar y crear un fichero MEX a partir de lo que se va a llamar un fichero C++-MEX (un fichero C++ que cumple las condiciones necesarias para poder crear con él un fichero MEX) es la siguiente:

```
>> mex nombre_archivo.cpp
```

donde nombre_archivo.cpp es el nombre del fichero en C++ y nombre_archivo.mexglx es el nombre del fichero MEX que se va a generar.

El código fuente de un fichero MEX programado en C++ tiene dos partes. La primera parte contiene el código de la función C++ que se quiere implementar como fichero MEX. La segunda parte es la función mexFunction que hace de interfaz entre C++ y MATLAB y que sustituye al main de C++. Los ficheros MEX deben incluir la librería "mex.h" donde está declarada la función mexFunction, que tiene la siguiente cabecera:

```
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *
    prhs[])
```

Los cuatro argumentos tienen los siguientes significados:

1. prhs es un vector de punteros a los valores de los argumentos de entrada ("right hand side arguments") que se van a pasar a la función C++.
2. nhrs es el número de argumentos de entrada de la función.
3. plhs y nlhs son análogos a los anteriores pero referidos a los argumentos de salida (left hand side arguments).

La función MEX trabaja con matrices, independientemente del tipo de dato a almacenar. Son las denominadas mxArrays.

Las funciones MEX tienen una extensión diferente en función de los sistemas operativos en que hayan sido generadas. Tanto $*.dll$ como $*.mexw32$ son para Windows. La extension $*.dll$ es para versiones de MATLAB anteriores a la 7.1. En Linux se usa la extensión $*.mexglx$.

5.2 Diseño del Software

En este apartado se van a explicar cada una de las funciones y scripts programados, esto es, las funciones en C++ (funciones MEX) que entrenan la máquina así como otras escritas en MATLAB que por su sencillez se ha optado por programarlas directamente en ese lenguaje.

5.2.1 LIBSVM

En el presente proyecto se han utilizado cuatro de las funciones que ofrece LIBSVM [7]:

libsvmwrite.c Es una función que escribe matrices de MATLAB a un archivo en formato LIBSVM:

```
>>libsvmwrite('data.txt', yo, X];
```

libsvmread.c Es una función que lee archivo en formato LIBSVM:

```
>>[yo, X] = libsvmread('data.txt');
```

Devuelve el vector de clases *yo* y los datos de entrenamiento X, el cual puede ser usado como datos de entrada para **svmtrain** o **svmpredic**.

svmtrain.c Es una función que realiza el entrenamiento de unos datos en función de unos parámetros. Desde general.m se llama a esta función de la siguiente manera:

```
>>model = svmtrain(yo, X,'-s 0 -c 10 -t 1 -g 0.02 -m 1000');
```

y en el terminal de ubuntu se hace con la siguiente linea de comando:

```
$ ./svm-train -s 0 -c 10 -t 1 -g 0.02 -m 1000 nombre.datos.entrenamientos
```

Como vemos, además de los datos de entrenamiento y sus respectivas etiquetas, hay una serie de parámetros configurables. En nuestro caso se han utilizado:

- $-s$ 0 indica que es un problema de SVM de tipo C-SVC.
- $-t$ 2 indica la utilización de un Kernel de tipo gaussiano.
- $-g$ 0.02 hace referencia al grado del Kernel gaussiano, γ. Si no se especifica por defecto γ toma el valor de 1 entre el el número de características (dimensión).
- $-c$ 10 valor del parámetro C, en este caso 10.
- $-m$ configura el tamaño de la memoria cache en 1000 Mb. (por defecto es 100)

La función devuelve model, que es un conjunto de datos con los resultados del entrenamiento, entre los cuales esta w. Para nuestro propuesta se modificó la función principal de LIBSVM para insertar el código de bajo costo cuya función llamaremos como svmtrain_mpi.

El esquema general del algoritmo de bajo costo, para entrenar una MVS, no hace el paso predictor ya que para problemas reales con esta modificación se logró convergencia del método en todos los problemas utilizados en esta investigación. En esta investigación se le hará referencia como algoritmo de bajo costo.

svmpredict.c Esta otra función de LIBSVM se encarga, una vez realizado el entrenamiento, de clasificar un conjunto de muestras. Desde general.m se le llama de la siguiente manera:

```
>>[predict_label, accuracy, dec_values] = svmpredict(yo_test,X_test
    , model);
```

La función recibe por parámetro los datos a clasificar y el modelo obtenido en el entrenamiento. Devuelve la predicción de la clase para cada muestra, la precisión y el porcentaje de datos bien clasificados.

En el sitio web de LIBSVM además de poder adquirir el software para diferentes entornos, se proporcionan manuales y otras herramientas para las máquinas de vectores de soporte.

5.2.2 buildQ.m

Este script se usa para calcular en el entorno de MATLAB, la matriz Hessiano de los problemas de programación cuadrática que surgen de las MVS. Emplea solo tres tipos de Kernel: Lineal, Polinomial y Gaussiano que son suficientes para esta investigación. Sin embargo genera limitaciones de memoria y tiempo para aquellos problemas que poseen un número muy grande de datos de entrenamiento.

5.2.3 lspect.m

Este script se usa para calcular la longitud espectral en el entorno de MATLAB. Esta función es llamada desde todos los ficheros de bajo costo.

5.2.4 Bajo_costo_con_identificacion.m

Este script se usa para identificar un conjunto de índices de patrones o variables candidatas a ser vectores de soporte, para después resolver un subproblema más pequeño por medio de la función svmtrain_mpi.

5.2.5 general_qp.m

Una vez que ya tenemos las funciones MEX compiladas y la constante C seleccionada, con este script, los datos son cargados y podemos comenzar el entrenamiento de la MVS y la clasificación de muestras. Esto es realizado por una serie de funciones que son llamadas desde general_qp.m, el cual llamaremos algoritmo de identificación de variables y cuya cabecera es:

```
ker = input(prompt); % Kernel lineal 0, polinomial 1, gaussiano 2.
c = input(prompt);   % C parametro del modelo

% Leer el archivo de datos

[yo, X] = libsvmread(datatrain);

% ns: numero de datos, dim: numero de caracteristicas
[ns,dim]= size(X);

% Construir el Kernel

[H]=buildQ(yo',X',ker);

e = ones(ns,1);

% Punto inicial

if (c > 1 || c == 1)
    xo = e*c; to = 1./(c*e); so = 0;
```

```
else
    xo = e*c; to = xo; so = 0;
end

% Selecci n del conjunto de  ndice con:
[x1,iter1,f1,x_select]=bajo_costo_con_identificacion(ns,xo,so,to,to,xo,
    yo,H,0,e,e*c);

if x_select ~= 0
libsvmwrite('data_select', yo(x_select), X(x_select,:));
[yo1, X1] = libsvmread('data_select');

% Se resuelve el subproblema con el algoritmo de bajo costo
model = svmtrain_mpi(yo1, X1,param);
[label_vector_test, instance_matrix_test] = libsvmread(datatest);
[predict_label, accuracy, dec_values] = svmpredict(label_vector_test,
    instance_matrix_test, model);

else
fprintf('\n No se hizo la identificacion, optimizacion finalizada');
end
```

CAPÍTULO VI
EXPERIMENTACIÓN NUMÉRICA

En este capítulo detallamos el marco experimental en el cual realizamos la experimentación númerica con los algoritmos que proponemos. Los apartados siguientes detallan:

- Los algoritmos utilizados como referencia para realizar comparaciones y los paquetes de software que implementan dichos algoritmos.
- Los conjuntos de datos utilizados en la experimentación.
- La metodología de experimentación y los parámetros utilizados.
- Los resultados numéricos de los experimentos realizados con los algoritmos de referencia.

6.1 Selección de Algoritmos de Referencia

Se seleccionaron dos algoritmos como base de comparación para los algoritmos que proponemos. De este modo, tener una idea más clara de las características de nuestra propuesta.

Los algoritmos seleccionados para la comparación incluyen los siguientes:

- Se utiliza como algoritmo principal para la comparación la implementación del algoritmo ASL introducido por Gonzalez-Lima, Hager, and Zhang (2011). La

versión usada para las comparaciones fue **ASL-1.1**, este código libre se puede descargar de la página *www.math.ufl.edu/∼hager/papers/CG/Archive/*.

- También se utilizará de LIBSVM, la herramienta que entrena la MVS llamado svmtrain, ya que es una implementación de las MVS muy rápida y robusta. Además, esta implementación es capaz de trabajar con más de dos clases. Su código fuente se puede descargar de *www.csie.ntu.edu.tw/∼cjlin/libsvm/*; la version usada fue **libsvm-3.14**.

6.2 Selección de los Problemas Artificiales

Se han utilizado algunos conjuntos de datos generados artificialmente para comprobar el rendimiento del algoritmo y obtener una primera impresión del comportamiento del mismo al realizar el ajuste de sus diversos parámetros. Los problemas artificiales son multidimensionales ya que muchos de los problemas reales por lo general son de dimension alta, sin embargo para poder ver su representación gráfica disponemos de cinco problemas, tres en $\mathbb{R}^2$ y dos en $\mathbb{R}^3$ para una compresión de los tipo de máquinas. Los datos que caracterizan dichos problemas se muestran en la Tabla 6–1.

Tabla 6–1: Problemas artificiales utilizados en los experimentos

Nombre	# Datos Entrenamiento	Dimension	Distribución de clases	% de datos mal clasificados
QP1	1000	2	500 / 500	0%
QP2	1000	3	500 / 500	0%
QP3	1000	2	524 / 476	1%
QP4	1000	3	517 / 483	1%
QP5	100	2	40 / 60	1%

Estos problemas sólo se utilizan para obtener una representación gráfica del tipo de soluciones que encuentra el algoritmo.

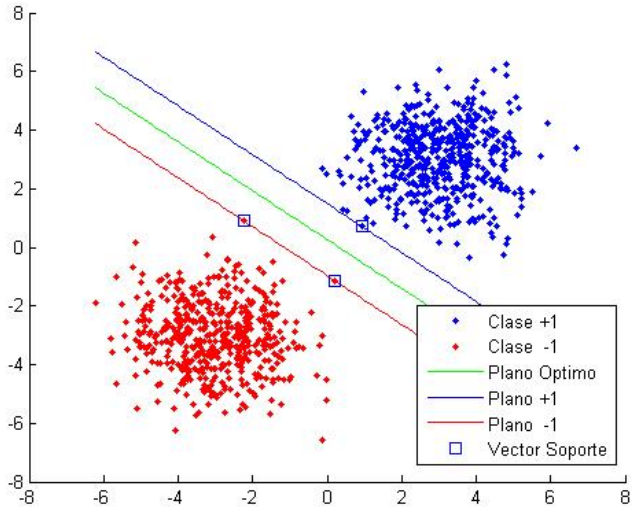

Figura 6–1: Conjuntos de datos que son linealmente separables en $\mathbb{R}^2$, problema QP1

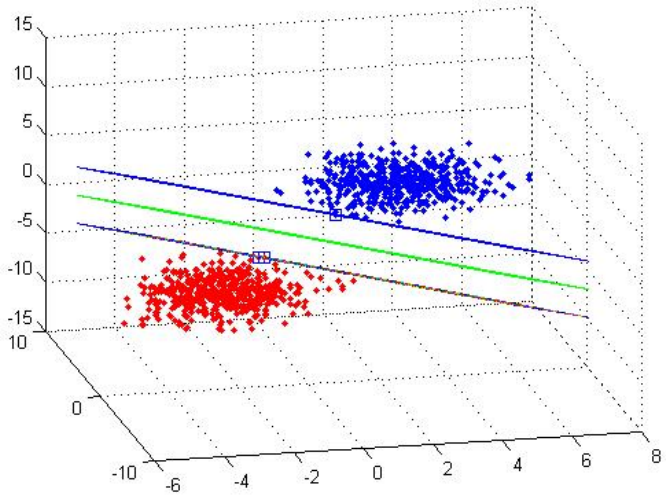

Figura 6–2: Conjuntos de datos que son linealmente separables en $\mathbb{R}^3$, problema QP2

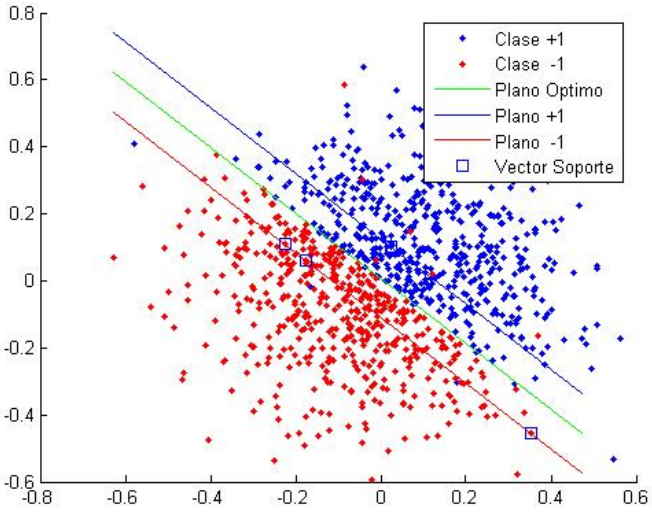

Figura 6–3: Conjuntos de datos que son linealmente no separables en $\mathbb{R}^2$, problema QP3

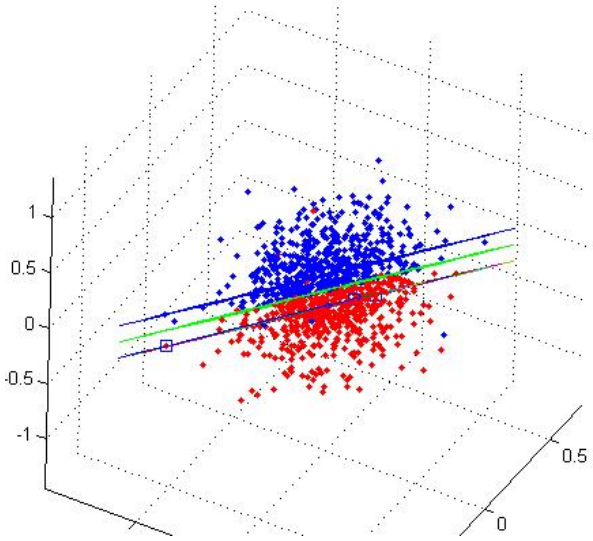

Figura 6–4: Conjuntos de datos que son linealmente no separables en $\mathbb{R}^3$, problema QP4

Por último mostraremos como la máquina de vectores clasifica cuando los datos no pueden ser separados por un hiperplano en su espacio de entrada original.

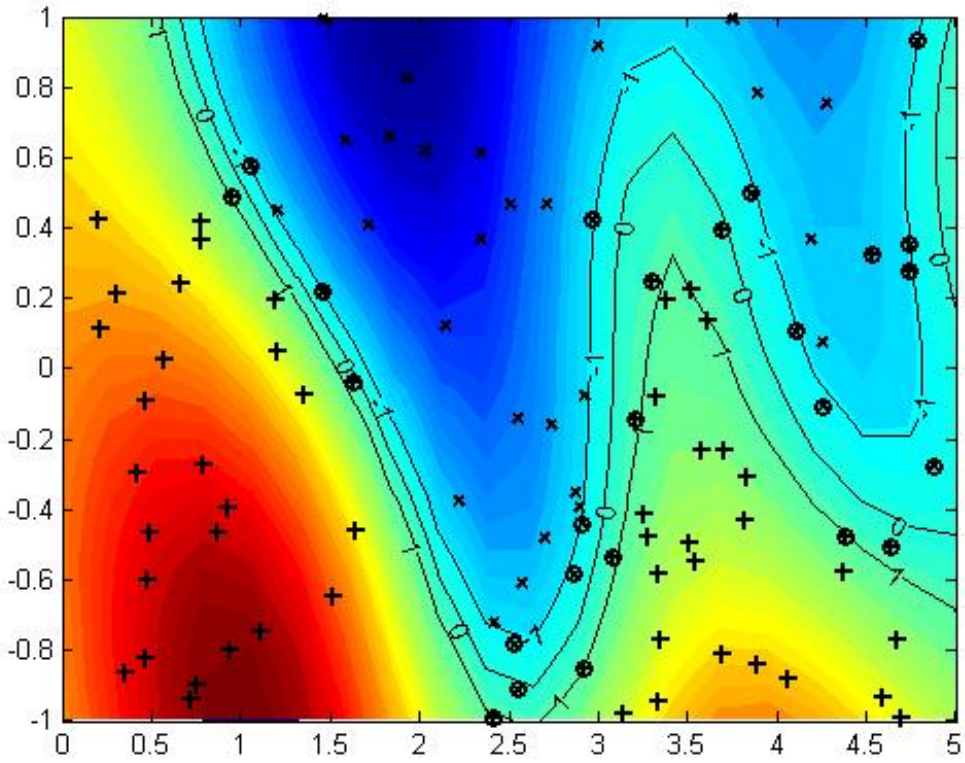

Figura 6–5: Conjunto de datos no lineales en $\mathbb{R}^2$, problema QP5

A continuación se mostrarán algunos resultados obtenidos en MATLAB, donde se comparan los tiempos y número de iteraciones que toman el método primal-dual de puntos interiores tradicional y nuestra propuesta de Bajo Costo predictor-corrector, en la resolución de problemas del tipo MVS, generados artificialmente para cada tipo de Kernel (Lineal, Polinomial y Gaussiano). Para cada problema se consideraron dos valores del parámetro C. En la segunda columna se muestran los números de datos considerados (n). Cada uno de estos datos z_i están en $\mathbb{R}^5$ y pertenece a una de las dos clases $y_i \in \{-1, 1\}$. Para cada método se usaron los mismos criterios de parada y obtuvieron las mismas soluciones con un porcentaje

igual a 1% de datos mal clasificados.

Tabla 6–2: Comparación del método primal-dual tradicional y el método propuesto de Bajo Costo en problemas aleatorios tipo MVS

	Datos para		Método Primal-dual			Método de Bajo Costo			
QP	entrenar (n)	C	iter	F.obj	seg	iter	F.obj	seg	Kernel
1	6000	1	-	-	-	450	-1042.8194	37.65	Lineal
		10	-	-	-	1541	-7846.5159	127.24	Lineal
2	6000	1	-	-	-	416	-1036.3671	34.36	Polinomial
		10	-	-	-	2007	-7510.7653	165.27	Polinomial
3	6000	1	-	-	-	297	-3110.9125	24.12	Gaussiano
		10	-	-	-	1380	-16516.0751	111.68	Gaussiano
4	5000	1	23	-910.3692	553.33	244	-910.3695	19.58	Lineal
		10	31	-6641.8403	749.01	3024	-6641.8408	195.79	Lineal
5	5000	1	21	-900.3986	512.32	368	-900.3987	22.08	Polinomial
		10	28	-6469.5801	668.72	2460	-6469.5801	144.17	Polinomial
6	5000	1	14	-2681.5757	519.92	397	-2681.5757	23.10	Gaussiano
		10	25	-14355.1190	937.58	1199	-14355.1191	69.70	Gaussiano
7	4000	1	20	-775.7444	257.65	193	-775.7445	8.57	Lineal
		10	32	-5582.6426	412.37	1766	-5582.6427	79.19	Lineal
8	4000	1	21	-746.0093	270.41	414	-746.0094	17.51	Polinomial
		10	29	-5159.9274	372.91	1288	-5159.9276	55.23	Polinomial
9	4000	1	15	-2248.8871	294.62	278	-2248.8871	11.75	Gaussiano
		10	21	-12571.0520	414.72	933	-12571.0565	39.83	Gaussiano

La Tabla 6–2 muestra que el método primal-dual tradicional no logra resolver los problemas generados 1,2 y 3; el símbolo "-" significa que en MATLAB no se encontró más memoria para las nuevas variables. Se aprecia que el método propuesto es más rápido, aunque requiere más iteraciones, que el método primal-dual tradicional.

6.3 Selección de Problemas Reales

Se ha seleccionado un conjunto de problemas de clasificación binaria a partir de la colección UCI y LIBSVM dados por los sitios *http://archive.ics.uci.edu.ml/* y *http://www.csie.ntu.edu.tw/∼cjlin/libsvm/* respectivamente para evaluar los algoritmos sobre datos reales.

Al escoger estos problemas se ha procurado que fuera un conjunto diverso en cuanto al número de atributos o dimensión. Igualmente hay tanto problemas con clases balanceadas como no balanceadas; todos escalados en el rango $[0, 1]$, y además se particionaron aquellos problemas que no tenían data test; un 80% para entrenar el clasificador y un 20% para test.

6.3.1 Condiciones de la Experimentación

Los experimentos llevados a cabo en este proyecto han sido realizados en el sistema operativo Ubuntu 12.04 LTS, en una computadora portátil HP Pavilon dv6 con un procesador i3 y 4GB de RAM. Las características de la computadora influirán directamente en el tiempo empleado por cada programa. El entrenamiento de una función de clasificación consiste en el ajuste de ciertos parámetros particulares del modelo utilizado de modo que éste represente una aproximación óptima de la función subyacente en los datos de entrenamiento. Sin embargo solo vamos a tomar como parámetros los siguientes valores dado el tipo de Kernel:

Lineal $K(z_i, z_j) = z_i^T z_j$.
Polinomial $K(z_i, z_j) = (\gamma z_i^T z_j)^3$.
Gaussiano $K(z_i, z_j) = exp(-\gamma \parallel z_i - z_j \parallel_2^2)$

donde γ es igual a 1/dimensión y el parámetro de regularización C que se usa en las restricciones de caja tomará los valores 1, 5 y 10, los parámetros relacionados con los algoritmos usados están descritos en los algoritmos del capítulo IV. Los conjuntos

de datos utilizados en los experimentos se pueden ver en la Tabla 6–3, junto con una breve descripción de sus atributos:

Tabla 6–3: Conjunto de problemas reales con sus atributos

Problema	Nro. datos de entrenamiento	Nro. datos test	Dimensión	Grupo
adwords1	52	12	4702	A
adwords2	52	12	3721	A
adwords3	52	12	242	A
adwords4	52	12	229	A
arsene	80	19	10000	B
duke	29	7	7129	B
leukemia	38	34	7129	B
colon-cancer	40	10	2000	B
farm-ads	3315	828	54877	C
news20	15997	3999	1355191	C
rcv1	20242	677399	47236	C
dorothea	641	160	937	C
dexter	240	60	19999	C
a1a	1605	30956	123	D
a2a	2265	30296	123	D
a3a	3185	29376	123	D
a4a	4781	27780	123	D
heart	216	54	13	E
ionosphere	281	70	34	E
liver-disorders	276	69	6	E
sonar	167	41	60	E
australian	552	138	14	F
breast-cancer	547	136	10	F
diabetes	615	153	8	F
fourclass	690	172	2	F
splice	1000	2175	60	G
svmguide1	3089	4000	4	G
svmguide3	1243	41	21	G
german.numer	800	200	24	G
w1a	2477	47272	300	H
w2a	3470	46279	300	H
w3a	4912	44837	300	H
gisette	6000	1000	5000	H
mushrooms	6500	1624	112	H

Los problemas mencionados en la Tabla 6–3 los clasificamos en 8 grupos, donde los primeras tres grupos A,B y C se refiere a aquellos problemas que poseen una alta dimensionalidad, los problemas de los grupos D,G y H son aquellos problemas de tamaño mediano donde la dimensionalidad es menor que el número de datos de entrenamientos y los grupos E y F son considerados problemas pequeños.

6.3.2 Comparaciones entre ASL, LIBSVM y Bajo Costo sin Paso Predictor

En este apartado se va a realizar una comparación entre el tiempo de ejecución, el valor de la función objetivo y el número de vectores de soporte representado por N_{VS} para cada uno de los kernel y diferentes valores del parámetro C de los tres algoritmos nombrados en la segunda linea de cada tabla.

Los resultados fueron obtenidos a través del terminal de ubuntu, ejecutando el algoritmo en C/C++, además, los resultados dados por ASL fueron proporcionados resolviendo el problema en su espacio original sin hacer descomposición del problema como lo hace LIBSVM, pero a pesar de no hacer descomposición, ASL tiene una estructura optimizada para resolver problemas con Kernel lineal que no se logró desactivar.

En la Tabla 6–4 se muestran los problemas de la clase A, que se describen en el lado izquierdo de la tabla, debajo del nombre del problema están reflejado el número de datos de entrenamiento al lado izquierdo y la dimensionalidad de los datos al lado derecho. Para este conjunto de problemas el algoritmo ASL supera en tiempo en la mayoría de los problemas. Por otro lado los algoritmos LIBSVM y Bajo Costo se mantuvieron a un mismo nivel de tiempo.

Tabla 6–4: Comparaciones entre ASL, LIBSVM y Bajo Costo con el grupo A

Problema			**Tiempo seg**			**Función Objetivo**			N_{sv}		
adwords1	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.004	0.01	0.01	-2.09771	-2.09771	-2.09771	50	50	50
		5	0.01	0.01	0.01	-10.09772	-10.09771	-10.09771	50	50	50
datos/dimension		10	0.01	0.01	0.01	-20.09772	-20.09771	-20.09771	50	50	50
52/4702	Polinomial	1	0.008	0.01	0.01	-45.99253	-45.99718	-45.28128	52	46	52
		5	0.008	0.01	0.01	-229.9323	-229.93405	-229.83172	52	48	52
		10	0.004	0.02	0.01	-459.7413	-459.74980	-459.75137	52	51	52
	Gaussiano	1	0.008	0.01	0.01	-43.01185	-43.01185	-43.01224	52	51	51
		5	0.008	0.01	0.01	-155.5269	-155.52690	-155.52699	52	51	51
		10	0.008	0.01	0.01	-213.2943	-213.29428	-213.29430	51	51	51
adwords2	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.004	0.02	0.01	-2.09677	-2.09677	-2.09677	49	48	49
		5	0.008	0.01	0.01	-10.09677	-10.09677	-10.09677	49	48	49
		10	0.01	0.01	0.01	-20.09677	-20.09677	-20.09677	49	48	49
52/3721	Polinomial	1	0.004	0.01	0.01	-45.98956	-45.99396	-45.80754	52	46	52
		5	0.008	0.01	0.01	-229.8631	-229.86516	-229.86641	52	50	52
		10	0.004	0.02	0.01	-459.4637	-459.46234	-459.52243	52	50	52
	Gaussiano	1	0.008	0.01	0.01	-42.06316	-42.06315	-42.06466	52	50	51
		5	0.008	0.01	0.01	-136.2885	-136.28848	-136.28860	52	51	51
		10	0.008	0.02	0.01	-180.4105	-180.41049	-180.41050	51	50	51
adwords3	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.001	0.01	0.01	-4.16999	-4.16999	-4.16999	47	46	47
		5	0.001	0.01	0.01	-47.45871	-4.74587	-4.74587	47	46	47
		10	0.001	0.01	0.01	-4.74586	-4.74587	-4.74587	47	46	47
52/242	Polinomial	1	0.001	0.01	0.01	-45.99471	-45.99931	-45.28354	52	46	52
		5	0.001	0.01	0.01	-229.9842	-229.98277	-229.88635	52	46	52
		10	0.001	0.01	0.01	-459.9367	-459.93111	-459.83198	52	46	52
	Gaussiano	1	0.001	0.01	0.01	-43.82652	-43.82652	-43.82670	51	51	51
		5	0.001	0.01	0.01	-175.6634	-175.66343	-175.66359	51	51	51
		10	0.001	0.01	0.01	-257.1658	-257.16582	-257.16584	51	50	51
adwords4	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.001	0.01	0.01	-4.247644	-4.24764	-4.24798	47	46	47
		5	0.001	0.01	0.01	-4.83139	-4.83139	-4.83139	48	47	48
		10	0.001	0.01	0.01	-4.83139	-4.83139	-4.83139	48	47	48
52/229	Polinomial	1	0.001	0.01	0.01	-45.9946	-45.99924	-45.28347	42	46	52
		5	0.001	0.01	0.01	-229.9825	-229.98115	-229.88464	52	46	52
		10	0.001	0.01	0.01	-459.930	-459.92463	-459.82521	52	46	52
	Gaussiano	1	0.001	0.01	0.01	-43.65135	-43.65136	-43.63240	52	49	49
		5	0.001	0.01	0.01	-171.2842	-171.28422	-171.28377	52	49	50
		10	0.001	0.01	0.01	-246.0747	-246.07467	-246.07469	49	49	49

En los resultados de la Tabla 6–5 los tres algoritmos tienen comportamiento comparable en el tiempo de ejecución, sin embargo, cabe destacar que con el problema **arsene** el valor de la función objetivo con los Kernel Polinomial y Gaussiano de ASL se mantuvieron por encima de los alcanzados por LIBSVM y Bajo Costo.

Tabla 6–5: Comparaciones entre ASL, LIBSVM y Bajo Costo con el grupo B

Problema			**Tiempo seg**			**Función Objetivo**			N_{sv}		
arsene	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.22	0.24	0.20	-0.03339	-0.03432	-0.03432	75	76	76
		5	0.23	0.24	0.19	-0.03339	-0.03432	-0.03432	75	76	76
datos/dimension		10	0.21	0.22	0.19	-0.03339	-0.03432	-0.03432	75	76	76
80/10000	Polinomial	1	0.18	0.21	0.19	-61.48001	-63.2470	-63.24705	76	77	77
		5	0.18	0.20	0.19	-204.4930	-210.7775	-210.77751	77	79	79
		10	0.18	0.20	0.19	-266.3341	-273.3735	-273.37357	78	79	79
	Gaussiano	1	0.20	0.19	0.19	-55.71258	-58.2016	-58.20166	80	75	75
		5	0.20	0.20	0.19	-105.6260	-161.8259	-161.82598	80	75	78
		10	0.20	0.20	0.18	-105.8061	-181.6816	-181.68163	80	77	77
duke	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.02	0.02	0.02	-0.0031	-0.0031	-0.0031	27	27	27
		5	0.02	0.01	0.02	-0.0031	-0.0031	-0.0031	27	27	27
		10	0.02	0.02	0.02	-0.0031	-0.0031	-0.0031	27	27	27
29/7129	Polinomial	1	0.01	0.02	0.01	-20.90789	-20.90788	-20.90788	29	29	29
		5	0.02	0.01	0.02	-69.50874	-69.50874	-69.50874	29	29	29
		10	0.02	0.01	0.02	-95.41262	-95.41261	-95.41490	27	27	29
	Gaussiano	1	0.02	0.02	0.01	-16.58519	-16.58519	-16.58520	29	29	29
		5	0.02	0.02	0.01	-19.10041	-19.10040	-19.10040	29	29	29
		10	0.02	0.02	0.01	-19.10041	-19.10040	-19.10040	29	29	29
leukemia	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.02	0.03	0.02	-0.0016	-0.0016	-0.0016	29	29	27
		5	0.02	0.03	0.03	-0.0016	-0.0016	-0.0016	29	29	27
		10	0.02	0.03	0.03	-0.0016	-0.0016	-0.0016	29	29	27
38/7129	Polinomial	1	0.02	0.03	0.03	-14.5796	-14.5796	-14.58	35	35	35
		5	0.02	0.03	0.03	-29.593	-29.5930	-29.5930	36	36	36
		10	0.02	0.03	0.03	-30.7646	-30.7646	-30.7646	37	37	37
	Gaussiano	1	0.03	0.02	0.03	-12.5254	-12.5254	-12.5254	38	38	38
		5	0.02	0.02	0.02	-13.8254	-13.8254	-13.8254	38	38	38
		10	0.03	0.02	0.03	-13.8254	-13.8254	-13.8254	38	38	38
colon-cancer	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Linear	1	0.008	0.01	0.01	-0.01026	-0.01026	-0.01026	26	26	26
		5	0.01	0.01	0.01	-0.01026	-0.01026	-0.01026	26	26	26
		10	0.01	0.01	0.01	-0.01026	-0.01026	-0.01026	26	26	26
40/2000	Polinomial	1	0.008	0.01	0.01	-15.55241	-15.55241	-15.55241	39	39	39
		5	0.008	0.01	0.01	-20.25473	-20.25473	-20.25473	38	38	38
		10	0.01	0.01	0.01	-20.25473	-20.25473	-20.25473	38	38	38
	Gaussiano	1	0.01	0.01	0.01	-15.39198	-15.39199	-15.39198	39	38	38
		5	0.008	0.01	0.02	-17.30594	-17.30593	-17.30593	38	38	38
		10	0.01	0.02	0.01	-17.30594	-17.30593	-17.30591	38	38	38

En la Tabla 6–6 donde se indique el valor de la celda con un "-" en las columnas del algoritmo ASL significa que no se resolvió el problema y el código no especifica por qué. También se puede decir que el algoritmo de Bajo Costo se comporta mejor en tiempo cuando se usan los Kernels Polinomial y Gaussiano.

Tabla 6–6: Comparaciones entre ASL, LIBSVM y Bajo Costo con el grupo C

Problema			**Tiempo seg**			**Función Objetivo**			N_{sv}		
Farm-ads	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	-	7.10	74.65	-	-28.75563	-28.75503	-	1207	1383
		5	-	7.14	227.38	-	-52.75670	-52.75670	-	1205	1381
datos/dimension		10	-	7.10	299.00	-	-82.75670	-82.75670	-	1206	1380
3315/54877	Polinomial	1	16.89	17.88	19.88	-3091.657	-3091.99911	-3080.49442	3315	3092	3315
		5	16.90	17.85	20.82	-15459.93	-15459.97780	-15456.96504	3315	3092	3315
		10	16.93	17.85	20.83	-30919.74	-30919.91121	-30916.62050	3315	3092	3315
	Gaussiano	1	18.54	18.87	22.41	-2898.774	-2898.72887	-2898.15122	3117	3104	3117
		5	18.70	16.47	25.22	-11413.49	-11412.63226	-11412.61865	2824	2787	2825
		10	18.83	14.25	27.66	-19600.80	-19598.63926	-19598.61880	2520	2484	2521
news20	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	-	363.87	1014.20	-	-2236.82221	-2236.88094	-	7463	7610
		5	-	354.84	1318.00	-	-2670.35155	-2670.34216	-	7292	7486
		10	-	345.99	1392.33	-	-2910.80809	-2910.75525	-	7165	7357
15997/1355191	Polinomial	1	561.16	882.86	1059.02	-15991.40	-15994	-15612.94606	15997	15994	15997
		5	526.15	818.40	902.89	-79970	-79970	-79985	15997	15994	15997
		10	565.71	819.59	880.12	-159940	-159940	-159970	15997	15994	15997
	Gaussiano	1	798.52	861.28	865.74	-15996.27	-15993.69995	-15996.70049	15997	15994	15997
		5	823.69	942.68	885.61	-80026.64	-79962.49883	-79977.49884	15996	15994	15997
		10	844.05	849.77	1046.80	-160266.6	-159910.02320	-159939.98685	15996	15994	15997
rcv1	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	-	89.83	647.8	-	-1745.6672	-1745.6799	-	4695	4819
		5	-	82.42	896.71	-	-2761.2710	-2761.2634	-	4157	4257
		10	-	80.92	944.40	-	-3159.2706	-3159.2479	-	4077	4165
20242/47236	Polinomial	1	279.15	361.20	383.90	-19499.73	-19502	-19466.4393	20242	19502	20242
		5	265.20	362.32	396.88	-97510	-97510	-97493.9755	20242	19502	20242
		10	267.83	362.58	397.62	-195020	-195020	-195011.5674	20242	19502	20242
	Gaussiano	1	336.56	408.72	439.15	-19491.29	-19476.4536	-19473.9519	20150	19502	19510
		5	339.77	409	451.52	-97242.58	-96871.3574	-96866.8803	19620	19502	19506
		10	376.11	412.32	440.15	-193950.4	-192465.4299	-192462.6565	19582	19502	19508
dorothea	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.74	0.54	1.93	-3.95944	-3.95944	-3.95944	225	226	225
		5	0.70	0.54	2.10	-3.95944	-3.95944	-3.95944	225	226	225
		10	0.70	0.53	2.09	-3.95944	-3.95944	-3.95944	225	226	225
641/937	Polinomial	1	1.15	0.76	1.21	-121.7155	-121.71543	-121.71560	410	388	399
		5	1.10	1.00	1.23	-549.4944	-549.49425	-549.49442	405	402	404
		10	1.18	1.06	1.31	-998.2728	-998.27271	-998.27264	405	401	403
	Gaussiano	1	1.14	0.56	1.30	-114.8693	-114.87280	-114.87284	282	280	281
		5	1.35	0.59	1.34	-458.5784	-458.66558	-458.66563	294	297	299
		10	1.36	0.61	1.38	-682.4241	-682.79684	-682.79689	301	302	302
dexter	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.1	1.29	1.22	-0.56826	-0.56826	-0.56826	237	237	237
		5	0.1	1.30	1.23	-0.56826	-0.56826	-0.56826	237	237	237
		10	0.1	1.27	1.14	-0.56826	-0.56826	-0.56826	237	237	237
240/19999	Polinomial	1	1.3	1.26	1.14	-239.4310	-239.43098	-239.43203	240	240	240
		5	1.33	1.00	1.13	-1185.774	-1185.77456	-1185.78071	240	240	240
		10	1.3	1.26	1.13	-2343.082	-2343.09825	-2343.10883	240	240	240
	Gaussiano	1	1.52	1.26	1.14	-237.0892	-236.89383	-236.89470	240	240	240
		5	1.35	1.26	1.14	-1127.103	-1122.34579	-1122.35191	240	240	240
		10	1.4	1.26	1.14	-2108.941	-2089.38316	-2089.39353	240	240	240

En las Tablas 6–7, 6–10 y 6–11, se pueden observar con mayor claridad que el nivel de complejidad para resolver los problemas aumenta cuando C aumenta. El algoritmo de Bajo Costo se comporta mejor cuando no se usa el Kernel Lineal.

Tabla 6–7: Comparaciones entre ASL, LIBSVM y Bajo Costo con el grupo D

Problema			**Tiempo seg**			**Función Objetivo**			N_{sv}		
a1a	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.2	0.2	5.1	-540.5751	-540.5749	-540.5750	589	588	591
		5	0.8	0.38	9.48	-2621.458	-2621.4577	-2621.1949	574	573	577
datos/dimension		10	2.4	0.71	19.24	-5208.56	-5208.5561	-5208.5371	567	568	571
1605 /123	Polinomial	1	0.3	0.18	0.48	-788.576	-788.5742	-788.1114	831	804	834
		5	0.3	0.18	0.55	-3914.399	-3914.3975	-3914.3877	830	827	880
		10	0.3	0.19	0.56	-7757.598	-7757.5954	-7757.5626	830	827	830
	Gaussiano	1	0.5	0.3	0.91	-670.0214	-673.0313	-673.0314	754	754	754
		5	0.6	0.24	1.45	-2936.157	-2936.1561	-2936.1559	660	660	660
		10	0.7	0.25	1.73	-5593.634	-5593.6332	-5593.6277	645	645	645
a2a	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.4	0.43	12.45	-828.3095	-828.3093	-828.3211	879	880	882
		5	1.9	0.8	25.89	-4043.987	-4043.9860	-4043.4906	862	861	887
		10	4.05	1.44	50.31	-8050.87	-8050.8747	-8049.9694	863	865	879
2265/123	Polinomial	1	0.6	0.36	1.04	-1141.469	-1141.4656	-1141.4544	1196	1163	1822
		5	0.6	0.37	1.09	-5656.718	-5656.7148	-5656.4356	1192	1179	1189
		10	0.7	0.4	1.18	-11186.87	-11186.8679	-11186.5236	1192	1184	1190
	Gaussiano	1	1.1	0.5	1.88	-971.6676	-971.6675	-971.6679	1068	1067	1068
		5	1.3	0.5	2.9	-4372.33	-4372.3296	-4372.2610	955	955	965
		10	1.4	0.5	4.82	-8413.124	-8413.1230	-8413.1200	935	932	937
a3a	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.8	0.74	20	-1104.831	-1104.8311	-1104.7492	1163	1164	1184
		5	3.1	1.42	86.66	-5438.342	-5438.3416	-5438.2598	1148	1147	1154
		10	5.7	2.36	157.53	-10850.007	-10850.0711	-10849.6180	1146	1147	1155
3185/123	Polinomial	1	1.2	0.69	1.87	-1541.532	-1541.5292	-1537.8564	1591	1560	1605
		5	1.3	0.68	2.29	-7618.292	-7618.2898	-7618.2816	1591	1584	1593
		10	1.5	0.71	1.93	-15013.17	-15013.1665	-15002.5027	1591	1588	3158
	Gaussiano	1	2.3	0.95	3.23	-1270.051	-1270.0509	-1269.9644	1389	1388	1510
		5	2.6	0.92	6.8	-5757.295	-5757.2945	-5757.2943	1249	1249	1252
		10	2.9	0.91	9.31	-11133.89	-11133.8904	-11133.8831	1228	1227	1229
a4a	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	1.5	1.74	69.79	-1669.31	-1669.3099	-1669.2424	1736	1735	1756
		5	7.5	3.16	228.93	-8284.248	-8284.2482	-8283.9533	1723	1723	1736
		10	12.56	4.62	382.54	-16550.024	-16550.2319	-16549.2282	1715	1714	1740
4781/123	Polinomial	1	2.8	1.64	4.41	-2364.483	-2364.4797	-2360.0523	2434	2395	2431
		5	2.9	1.65	3.97	-11592.06	-11592.0609	-11549.0834	2433	2414	2473
		10	3.6	1.64	5.25	-22608.26	-22608.2554	-22608.2621	2433	2421	2585
	Gaussiano	1	5.3	2.14	9.27	-1855.879	-1855.8783	-1855.8769	2002	2000	2006
		5	6.2	2.08	18.26	-8550.435	-8550.4348	-8550.4346	1821	1819	1821
		10	7.9	2.09	30.75	-16610.04	-16610.0248	-16609.9876	1793	1793	1794

En las Tablas 6–8 y 6–9 donde se muestran los problemas pequeños, los tres algoritmos mostraron ser competitivos en el tiempo de ejecución.

Tabla 6–8: Comparaciones entre ASL, LIBSVM y Bajo Costo con el grupo E

Problema			**Tiempo seg**			**Función Objetivo**			N_{sv}		
heart	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.01	0.04	0.04	-76.9619	-76.9619	-76.9619	85	85	85
		5	0.04	0.01	0.12	-375.5676	-375.5672	-375.5672	82	82	82
datos/dimension		10	0.07	0.01	0.16	-748.6151	-748.6138	-748.6135	82	82	82
216/13	Polinomial	1	0.004	0.01	0.01	-111.9197	-111.9196	-111.9196	155	155	155
		5	0.008	0.01	0.02	-374.713	-374.7129	-374.7130	117	117	117
		10	0.1	0.01	0.02	-623.3628	-623.3627	-623.3628	110	110	110
	Gaussiano	1	0.008	0.01	0.01	-84.4169	-84.7413	-84.7471	117	116	116
		5	0.02	0.01	0.02	-304.6447	-320.3226	-320.3227	102	99	99
		10	0.03	0.01	0.04	-512.0749	-545.7628	-545.7628	95	100	100
ionosphere	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.03	0.01	0.1	-54.4542	-54.4542	-54.4542	78	78	78
		5	0.06	0.01	0.17	-194.6065	-194.6064	-194.6064	60	60	60
		10	0.08	0.02	0.21	-341.3126	-341.3123	-341.3120	57	57	57
281/34	Polinomial	1	0.01	0.01	0.02	-179.3521	-179.3521	-179.3520	205	205	205
		5	0.02	0.01	0.03	-628.8219	-623.8218	-623.8212	188	188	189
		10	0.1	0.01	0.04	-1020.019	-1020.0194	-1020.0194	168	168	168
	Gaussiano	1	0.02	0.01	0.04	-80.6734	-79.4820	-79.4820	120	116	116
		5	0.03	0.01	0.07	-200.76	-224.5100	-224.5101	79	83	83
		10	0.04	0.01	0.07	-286.6906	-336.2692	-336.2766	69	74	74
liver-disorders	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.01	0.01	0.02	-217.3969	-217.3969	-217.3963	230	230	230
		5	0.02	0.01	0.07	-1034.554	-1034.5543	-1034.5543	215	215	215
		10	0.02	0.01	0.1	-2043.119	-2043.1207	-2043.1206	210	210	210
276/6	Polinomial	1	0.008	0.01	0.01	-230.197	-230.1969	-230.1774	236	237	236
		5	0.004	0.01	0.01	-1114.924	-1114.9242	-1114.9224	236	236	237
		10	0.01	0.01	0.02	-2150.572	-2150.5722	-2150.5721	233	233	233
	Gaussiano	1	0.02	0.01	0.02	-224.303	-223.5044	-223.5044	237	237	237
		5	0.008	0.01	0.04	-1068.343	-1010.6287	-1010.6286	217	222	222
		10	0.03	0.01	0.06	-2179.503	-1911.6405	-1911.6387	209	216	217
sonar	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.01	0.01	0.03	-50.3928	-50.3927	-50.3927	76	76	76
		5	0.02	0.01	0.08	-162.5459	-162.5456	-162.5456	66	66	66
		10	0.05	0.02	0.08	-257.8122	-257.8118	-257.8117	64	64	64
167/60	Polinomial	1	0.004	0.01	0.01	-137.6318	-137.6317	-137.6322	157	157	157
		5	0.008	0.01	0.01	-510.4846	-510.4845	-510.4846	138	138	138
		10	0.008	0.01	0.01	-860.2321	-860.2321	-860.2321	124	124	124
	Gaussiano	1	0.008	0.01	0.01	-99.787	-99.7870	-99.7870	128	128	128
		5	0.01	0.01	0.02	-318.6368	-318.6367	-318.6400	102	102	102
		10	0.01	0.01	0.03	-475.0782	-475.0780	-475.0781	93	93	93

Tabla 6–9: Comparaciones entre ASL, LIBSVM y Bajo Costo con el grupo F

Problema			**Tiempo seg**			**Función Objetivo**			N_{sv}		
australian	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.4	0.03	1.53	-155.4755	-157.5403	-157.5362	165	165	178
		5	1.7	0.12	5.27	-776.2349	-783.5652	-783.5120	162	165	180
datos/dimension		10	3.1	0.17	7.36	-1545.433	-1566.0960	-1565.7829	162	166	189
552/14	Polinomial	1	0.04	0.01	0.11	-148.8364	-186.6941	-186.6972	241	245	245
		5	0.1	0.01	0.24	-749.097	-758.2775	-758.2774	212	209	210
		10	0.1	0.01	0.38	-1391.944	-1409.2541	-1409.2543	200	204	204
	Gaussiano	1	0.08	0.02	0.25	-161.9744	-161.0890	-161.0953	208	199	200
		5	0.1	0.02	0.39	-690.442	-721.2454	-721.2441	195	192	193
		10	0.1	0.02	0.59	-1302.842	-1359.5527	-1359.5497	184	194	194
breast-cancer	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.02	0.03	0.07	-40.1563	-40.1563	-40.1563	47	47	47
		5	0.08	0.02	0.15	-190.5168	-190.5168	-190.5167	43	43	43
		10	0.1	0.02	0.17	-377.4098	-377.4094	-377.4088	43	44	44
547/10	Polinomial	1	0.02	0.01	0.05	-52.3947	-52.3947	-52.3947	80	80	80
		5	0.04	0.01	0.06	-189.3929	-189.3928	-189.3928	55	54	55
		10	0.1	0.01	0.07	-383.5931	-337.9786	-337.9786	61	57	58
	Gaussiano	1	0.04	0.01	0.06	-46.6526	-46.6526	-46.6526	63	62	63
		5	0.06	0.01	0.1	-180.1388	-180.1387	-180.1387	55	54	55
		10	0.07	0.01	0.12	-325.8576	-325.8575	-325.8575	61	60	61
diabetes	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.04	0.01	0.2	-324.2372	-324.2370	-324.2370	336	336	337
		5	0.2	0.02	0.54	-1593.459	-1593.4569	-1593.4552	324	325	326
		10	0.2	0.03	0.9	-3179.164	-3179.1489	-3179.1441	324	324	325
615/8	Polinomial	1	0.02	0.02	0.03	-407.8198	-407.8198	-407.8184	435	432	434
		5	0.03	0.01	0.08	-1790087	-1790.0874	-1790.0854	396	396	396
		10	0.03	0.01	0.13	-3387.031	-3387.0312	-3387.0313	373	373	373
	Gaussiano	1	0.06	0.02	0.1	-335.5127	-335.9479	-335.9479	367	369	369
		5	0.08	0.04	0.23	-1533.107	-1535.0097	-1535.0097	328	328	328
		10	0.1	0.04	0.38	-2978.909	-2981.9030	-2981.9028	324	326	326
fourclass	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.03	0.01	0.18	-385.4049	-385.4049	-385.3990	389	389	392
		5	0.1	0.02	0.6	-1917.85	-1917.8499	-1917.8420	386	386	389
		10	0.02	0.02	1.04	-3833.368	-3833.3667	-3833.3653	385	386	386
690/2	Polinomial	1	0.04	0.02	0.17	-374.9067	-374.9066	-374.9066	407	407	407
		5	0.08	0.01	0.47	-1693.041	-1693.0399	-1693.0408	355	356	356
		10	0.1	0.01	0.5	-3292.884	-3292.8840	-3292.8462	344	344	348
	Gaussiano	1	0.1	0.02	0.28	-328.9492	-329.6443	-329.6439	349	351	353
		5	0.1	0.02	0.58	-1504.71	-1492.9650	-1492.9094	326	328	332
		10	0.2	0.02	0.87	-2819.634	-2788.5523	-2788.5400	318	320	320

Tabla 6–10: Comparaciones entre ASL, LIBSVM y Bajo Costo con el grupo G

Problema			**Tiempo seg**			**Función Objetivo**			N_{sv}		
splice	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.5	0.37	2.02	-375.1866	-375.1859	-375.1788	404	407	413
		5	3	1.11	6.79	-1864.575	-1864.5753	-1864.5230	408	408	412
datos/dimension		10	4.4	2.58	8.96	-3726.268	-3726.2680	-3725.7394	407	407	417
1000/60	Polinomial	1	0.1	0.16	0.2	-828.3506	-828.3506	-828.3691	996	996	996
		5	0.2	0.18	0.33	-2267.554	-2267.5544	-2267.5553	960	955	960
		10	0.3	0.19	0.27	-2743.695	-2743.6955	-2743.6954	937	928	937
	Gaussiano	1	0.3	0.14	0.5	-369.7973	-369.7972	-369.7938	607	607	614
		5	0.4	0.21	0.67	-712.6871	-712.6870	-712.6871	569	569	576
		10	0.47	0.22	0.71	-749.9003	-749.9003	-749.9102	587	587	595
svmguide1	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.2	0.08	3.74	-479.457	-479.4570	-479.4491	554	553	557
		5	0.3	0.08	7.48	-2028.024	-2028.0238	-2027.8779	434	434	444
		10	0.5	0.08	12.81	-3923.111	-3923.1111	-3923.0818	409	410	410
3089/4	Polinomial	1	0.4	0.09	1.71	-604.195	-604.1949	-604.1923	767	767	768
		5	0.6	0.09	3.19	-2106.932	-2106.9322	-2106.9322	504	505	504
		10	0.8	0.06	3.71	-3760.097	-3760.0970	-3760.0942	437	437	437
	Gaussiano	1	1.3	0.23	3.67	-507.3071	-507.3070	-507.3043	630	630	638
		5	1.5	0.17	6.47	-1855.325	-1855.3252	-1855.3242	434	435	435
		10	1.8	0.16	10.46	-3358.886	-3358.8859	-3358.8858	386	386	386
svmguide3	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.1	0.07	2.98	-525.1375	-525.1373	-525.1361	546	545	548
		5	0.5	0.07	7.77	-2554.995	-2554.9923	-2554.9686	529	530	534
		10	0.9	0.11	12.26	-5075.782	-5075.7756	-5075.7445	521	521	526
1243/21	Polinomial	1	0.1	0.06	0.56	-567.8805	-567.8795	-567.8933	589	582	594
		5	0.2	0.06	0.95	-2755.003	-2755.0024	-2754.9881	579	573	582
		10	0.3	0.05	1.32	-5434.621	-5434.6209	-5434.6230	576	573	576
	Gaussiano	1	0.3	0.13	1.12	-545.7283	-545.9010	-545.9009	572	573	573
		5	0.4	0.12	2.17	-2682.503	-2585.0912	-2585.0893	546	555	558
		10	0.4	0.12	2.63	-5472.639	-5000.8251	-5000.7995	515	535	537
german.numer	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.2	0.06	1.4	-414.6108	-414.6105	-414.5959	428	430	429
		5	0.6	0.13	3.85	-2066.244	-2066.2399	-2066.2340	426	427	428
		10	0.7	0.21	6.05	-4130.706	-4130.6920	-4130.6045	426	426	428
800/24	Polinomial	1	0.07	0.05	0.13	-415.6606	-415.6606	-415.6605	506	506	506
		5	0.1	0.07	0.27	-1595.188	-1595.1877	-1595.1835	475	475	475
		10	0.2	0.06	0.57	-2699.9	-2699.9002	-2699.9003	462	462	462
	Gaussiano	1	0.1	0.06	0.28	-409.7153	-405.8057	-405.8058	505	482	482
		5	0.3	0.07	0.45	-1491.872	-1611.8384	-1611.8384	478	461	461
		10	0.3	0.08	0.84	-2420.408	-2770.7516	-2770.7416	457	451	452

Tabla 6–11: Comparaciones entre ASL, LIBSVM y Bajo Costo con el grupo H

Problema			**Tiempo seg**			**Función Objetivo**			N_{sv}		
w1a	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.2	0.09	4.54	-62.9151	-62.9151	-62.9151	172	168	172
		5	0.5	0.13	11.68	-213.7925	-213.7924	-213.7788	161	155	170
datos/dimension		10	0.5	0.14	16.77	-383.775	-383.7749	-383.7750	400	159	386
2477/300	Polinomial	1	0.7	0.04	1.02	-143.9922	-143.9905	-143.9754	2091	146	2467
		5	0.8	0.06	8.74	-719.8096	-719.7997	-719.8115	1891	179	1277
		10	0.9	0.06	12.16	-1439.241	-1439.2100	-1438.9770	1974	193	1174
	Gaussiano	1	1.5	0.1	4.11	-140.8228	-140.8227	-140.8228	246	208	240
		5	1.7	0.13	4.58	-641.705	-641.7047	-641.7050	237	228	236
		10	1.5	0.13	4.42	-1179.252	-1179.2520	-1179.2518	215	210	217
w2a	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.3	0.16	9.43	-99.9954	-99.9953	-100.0042	230	222	229
		5	0.6	0.21	15.52	-359.379	-359.3790	-359.3789	207	201	207
		10	1.3	0.27	29.57	-633.2965	-633.2964	-633.2960	200	192	201
3470/300	Polinomial	1	1.3	0.09	2.15	-213.9859	-213.9827	-211.9931	2809	217	3058
		5	1.7	0.12	16.1	-1069.52	-1069.6421	-1069.6012	2389	269	1542
		10	2	0.12	11.94	-2138.615	-2138.58657	-2138.6206	2170	290	1708
	Gaussiano	1	2.8	0.21	6.7	-208.6953	-208.6951	-208.6931	295	274	291
		5	2.8	0.23	8.59	-944.9906	-944.9902	-944.9637	286	280	288
		10	3.1	0.24	8.97	-1738.319	-1738.3181	-1738.3097	292	288	295
w3a	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.6	0.28	22.33	-140.7621	-140.7620	-140.7722	279	275	281
		5	1.5	0.52	58.82	-523.4899	-523.4899	-523.4880	259	250	261
		10	4	0.74	93.45	-949.1873	-949.1866	-949.1667	251	244	253
4912/300	Polinomial	1	2.6	0.15	5.96	-285.982	-285.9789	-285.8901	4215	290	3100
		5	4.2	0.21	31.91	-1429.567	-1429.5514	-1429.5697	2850	353	2134
		10	4.7	0.24	67.09	-2858.273	-2858.2482	-2858.2727	2718	406	1757
	Gaussiano	1	5.6	0.38	13.93	-278.8692	-278.8689	-278.8700	367	353	365
		5	5.9	0.4	14.53	-1262.951	-1262.9509	-1262.9497	354	346	352
		10	6.2	0.4	16.66	-2323.916	-2323.9162	-2323.9085	351	340	354
gisette	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	16.25	94.75	530.16	-0.6680	-0.6680	-0.6679	1084	1084	1080
		5	17.95	95.43	580.65	-0.6680	-0.6680	-0.6679	1084	1084	1080
		10	17.25	104.27	581.50	-0.6680	-0.6680	-0.6679	1084	1084	1080
6000/5000	Polinomial	1	543.36	145.93	519.60	-775.3216	-775.32153	-775.3430	1631	1627	1627
		5	556.96	131.41	506.28	-1161.511	-1161.5109	-1161.5110	1464	1459	1462
		10	556.32	144	517.64	-1175.351	-1175.3512	-1175.3513	1487	1482	1484
	Gaussiano	1	573.80	169.42	505.55	-881.0845	-880.5489	-880.5602	1670	1666	1666
		5	577.66	141	493.55	-1500.151	-1499.3972	-1499.3974	1370	1362	1363
		10	584.09	140.14	495.16	-1558.630	-1557.9353	-1557.9356	1398	1400	1400
mushrooms	**Kernel**	**C**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**	**ASL**	**LIBSVM**	**Bajo Costo**
	Lineal	1	0.73	0.28	39.57	-7.5726	-7.5726	-7.5725	464	228	400
		5	0.76	0.29	78.26	-7.5726	-7.5726	-7.5723	415	231	505
		10	0.71	0.27	68.79	-7.5726	-7.5726	-7.5723	426	231	480
6500/112	Polinomial	1	7.74	4.54	36.77	-2115.434	-2115.4397	-2115.4573	3647	3336	3516
		5	10.31	2.2	45.22	-4184.787	-4184.7911	-4184.6859	1809	1581	1819
		10	10.98	1.63	47.64	-5091.134	-5091.1693	-5090.9426	1343	1122	1349
	Gaussiano	1	14.41	1.2	37.91	-273.0173	-273.0186	-273.0203	592	549	565
		5	15.48	0.76	60.05	-342.5207	-342.5229	-342.5247	318	280	292
		10	15.94	0.83	56.59	-357.8696	-357.8765	-357.7655	301	255	309

6.3.3 Comparaciones entre Bajo Costo sin Paso Predictor y Bajo Costo con Identificación de Variables

En esta sección se compara los tiempos de corrida, el número de vectores de soportes y el error de generalización (porcentaje de datos bien clasificados), entre el algoritmo de Bajo Costo e Identificación de Variables, para cada problema de la Tabla 6–3.

La idea de resolver los problemas usando identificación de variables, surge del hecho que en la mayoría de los problemas reales, pocos datos de entrenamiento lograrán ser vectores de soporte no acotados, mientras que una gran parte serán vectores de soporte acotados si son mal clasificados o cero si no aportan ningún valor a la función objetivo, sin embargo, se necesita verificar si al disminuir el número de muestras se sigue manteniendo el porcentaje de datos bien clasificados sobre los datos nuevos (datos test) obtenidos al resolver el problema general con el algoritmo de Bajo Costo. Es por esta razón que agregamos a las tablas de esta sección el error de clasificación de ambos algoritmos.

Cabe destacar que los valores de la funciones objetivos no se comparan en estas tablas, ya que el algoritmo de identificación de variables excluye aquellos datos de entrenamientos cuya variables dual está en las cotas ($x = 0$ ó $x = C$), por ende al resolver el subproblema con la variables seleccionadas, el máximo margen de separación entre las dos clases será afectado, y no sería el valor óptimo sino una aproximación.

Tabla 6–12: Comparaciones entre Bajo Costo e Identificación de Variable con el grupo A

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**	
adwords1	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.01	0.02	50	48	100%	100%
Nro. Datos		5	0.01	0.006	50	16	100%	100%
entrenamiento/test		10	0.01	0.16	50	48	100%	100%
52/12	Polinomial	1	0.01	0.003	52	18	50%	58.33%
		5	0.01	0.004	52	38	50%	50%
		10	0.01	0.005	52	32	50%	50%
	Gaussiano	1	0.01	0.006	51	29	50%	50%
		5	0.01	0.007	51	25	91.66%	66.66%
		10	0.01	0.01	51	37	100%	91.99%
adwords2	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.01	0.02	49	44	83.33%	83.33%
		5	0.01	0.01	49	12	83.33%	58.33%
		10	0.01	0.17	49	46	83.33%	83.33%
52/12	Polinomial	1	0.01	0.002	52	18	50%	50%
		5	0.01	0.01	52	39	50%	50%
		10	0.01	0.006	52	37	50%	50%
	Gaussiano	1	0.01	0.01	51	28	50%	50%
		5	0.01	0.01	51	32	75%	75%
		10	0.01	0.009	51	38	83.33%	83.33%
adwords3	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.01	0.007	47	42	83.33%	83.33%
		5	0.01	0.01	47	46	66.66%	66.66%
		10	0.01	0.03	47	47	66.66%	66.66%
52/12	Polinomial	1	0.01	0.004	52	18	50%	50%
		5	0.01	0.007	52	37	50%	50%
		10	0.01	0.004	52	37	50%	50%
	Gaussiano	1	0.01	0.004	51	27	50%	50%
		5	0.01	0.005	51	22	50%	50%
		10	0.01	0.01	51	33	91.66%	50%
adwords4	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.01	0.005	47	37	100%	100%
		5	0.01	0.01	48	48	100%	100%
		10	0.01	0.01	48	47	100%	100%
52/12	Polinomial	1	0.01	0.003	52	18	50%	50%
		5	0.01	0.004	52	38	50%	50%
		10	0.01	0.008	52	37	50%	50%
	Gaussiano	1	0.01	0.005	49	23	50%	50%
		5	0.01	0.003	50	26	58%	83.33%
		10	0.01	0.009	49	29	91%	83.33%

Tabla 6–13: Comparaciones entre Bajo Costo e Identificación de Variable con el grupo B

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**	
arsene	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.20	0.39	76	65	78.94%	78.94%
Nro. Datos		5	0.19	0.20	76	66	78.94%	78.94%
entrenamiento/test		10	0.19	0.14	76	54	78.94%	78.94%
80/19	Polinomial	1	0.19	0.02	77	16	63.15%	73.68%
		5	0.19	0.11	79	30	73.68%	57.89%
		10	0.19	0.19	79	68	89.47%	78.94%
	Gaussiano	1	0.19	0.04	75	18	57.89%	63.15%
		5	0.19	0.15	78	60	78.94%	68.42%
		10	0.18	0.21	77	77	84.21%	84.21%
duke	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.02	0.01	27	4	85.71%	57.14%
		5	0.02	0.01	27	4	85.71%	57.14%
		10	0.02	0.01	27	4	85.71%	85.71%
29/7	Polinomial	1	0.01	0.006	29	7	41.85%	42.85%
		5	0.02	0.02	29	19	42.85%	42.85%
		10	0.02	0.03	27	22	57.14%	57.14%
	Gaussiano	1	0.01	0.003	29	11	85.71%	57.14%
		5	0.01	0.02	29	29	85.71%	85.71%
		10	0.01	0.01	29	29	85.71%	85.71%
leukemia	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.02	0.01	27	6	82.35%	73.52%
		5	0.03	0.01	27	6	82.35%	76.47%
		10	0.03	0.02	27	6	82.35%	85.29%
38/34	Polinomial	1	0.03	0.01	35	22	58.82%	58.82%
		5	0.03	0.01	36	24	58.82%	58.82%
		10	0.03	0.02	37	29	58.82%	58.82%
	Gaussiano	1	0.03	0.02	38	29	67.64%	58.82%
		5	0.02	0.03	38	38	70.58%	70.58%
		10	0.03	0.03	38	38	70.58%	70.58%
colon-cancer	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.01	0.01	26	4	80%	50%
		5	0.01	0.02	26	4	80%	50%
		10	0.01	0.02	26	4	80%	50%
40/10	Polinomial	1	0.01	0.002	39	20	60%	60%
		5	0.01	0.01	38	36	60%	60%
		10	0.01	0.01	38	38	60%	60%
	Gaussiano	1	0.01	0.005	38	27	70%	60%
		5	0.02	0.01	38	37	80%	80%
		10	0.01	0.009	38	36	80%	80%

En aquellas tablas donde la columna asociada a identificación posea una celda en blanco significa que no se pudo generar la matriz Hessiana ya que por ser matrices de gran tamańo, no había memoria suficiente en MATLAB para guardarlas y en aquellos problemas donde el algoritmo no hace la identificación (encuentra el óptimo antes de hacer la identificación) se representa con un "-".

Tabla 6–14: Comparaciones entre Bajo Costo e Identificación de Variable con el grupo C

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**	
Farm-ads	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	74.65	61.71	1383	1346	87.31%	87.19%
Nro. Datos		5	227.38	maxIter	1381	maxIter	87.43%	maxIter
entrenamiento/test		10	299.00	135.25	1380	1251	87.43%	87.19%
3315/828	Polinomial	1	19.88	2.42	3315	958	53.26%	52.53%
		5	20.82	9.67	3315	2227	53.26%	53.26%
		10	20.83	12.94	3315	2261	53.26%	53.26%
	Gaussiano	1	22.41	2.11	3117	735	75.60%	53.26%
		5	25.22	8.98	2825	1059	75%	75.84%
		10	27.66	7.93	2521	1021	77.77%	75.48%
news20	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	1014.20		7610		97.04%	
		5	1318.00		7486		97.27%	
		10	1392.33		7357		97.14%	
15997/3999	Polinomial	1	1059.02		15997		49.98%	
		5	902.89		15997		90.17%	
		10	880.12		15997		90.17%	
	Gaussiano	1	865.74		15997		49.98%	
		5	885.61		15997		49.98%	
		10	1046.80		15997		49.98%	
rcv1	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	647.8		4819		96.32%	
		5	896.71		4257		95.65%	
		10	944.40		4165		95.43%	
20242/677399	Polinomial	1	383.90		20242		52.47%	
		5	396.88		20242		52.47%	
		10	397.62		20242		52.47%	
	Gaussiano	1	439.15		19510		47.52%	
		5	451.52		19506		47.52%	
		10	440.15		19508		47.52%	
dorothea	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	1.93	1.38	225	210	85.62%	85.62%
		5	2.10	1.45	225	220	85.62%	85.62%
		10	2.09	1.85	225	218	85.62%	85.62%
641/160	Polinomial	1	1.21	1.12	399	318	90%	90%
		5	1.23	0.25	404	251	90%	90%
		10	1.31	1.21	403	391	90.62%	90.62%
	Gaussiano	1	1.30	1.17	281	216	90%	90%
		5	1.34	0.48	299	226	90.62%	80.62%
		10	1.38	0.57	302	155	90%	90.62%
dexter	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	1.22	0.04	237	30	91.66%	53.33%
		5	1.23	0.06	237	36	91.66%	68.33%
		10	1.14	0.09	237	50	91.66%	65%
240/60	Polinomial	1	1.14	0.09	240	58	70%	61.66%
		5	1.13	0.01	240	20	70%	63.33%
		10	1.13	0.01	240	24	70%	61.66%
	Gaussiano	1	1.14	0.09	240	62	70%	61.66%
		5	1.14	0.01	240	28	70%	61.66%
		10	1.14	0.03	240	38	70%	61.66%

Tabla 6–15: Comparaciones entre Bajo Costo e Identificación de Variables con el grupo D

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**	
a1a	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	5.1	0.57	591	186	83.70%	83.91%
Nro. Datos		5	9.48	2.56	577	377	83.87%	83.29%
entrenamiento/test		10	19.24	3.58	571	338	83.76%	83.00%
1605/30956	Polinomial	1	0.48	-	834	-	75.94%	-
		5	0.55	0.5	880	660	75.94%	75.94%
		10	0.56	0.5	830	742	75.94%	75.94%
	Gaussiano	1	0.91	0.27	754	287	83.58%	83.55%
		5	1.45	0.43	660	173	84.32%	84.02%
		10	1.73	0.46	645	189	84.42%	83.70%
a2a	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	12.45	1.11	882	263	84.28%	84.44%
		5	25.89	8.11	887	541	84.02%	83.69%
		10	50.31	13.32	879	578	84.02%	83.28%
2265/30296	Polinomial	1	1.04	-	1822	-	76.00%	-
		5	1.09	0.67	1189	963	76.00%	76.00%
		10	1.18	0.75	1190	858	76.00%	76.00%
	Gaussiano	1	1.88	0.48	1068	297	83.93%	83.78%
		5	2.9	0.83	965	183	84.53%	84.32%
		10	4.82	0.91	937	182	84.64%	84.34%
a3a	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	20	3.54	1184	518	84.33%	84.33%
		5	86.66	16.54	1154	707	84.09%	83.81%
		10	157.53	36.21	1155	760	84.06%	83.79%
3185/39376	Polinomial	1	1.87	-	1605	-	75.93%	-
		5	2.29	0.97	1593	1422	75.93%	75.93%
		10	1.93	1.39	3158	1348	75.93%	75.93%
	Gaussiano	1	3.23	1.06	1510	377	83.84%	83.53%
		5	6.8	1.62	1252	210	84.48%	84.22%
		10	9.31	3.86	1229	1031	84.46%	84.43%
a4a	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	69.79	7.99	1756	665	84.27%	84.56%
		5	228.93	58.09	1736	1199	84.45%	84.42%
		10	382.54	73.81	1740	1151	80.03%	84.26%
4781/27780	Polinomial	1	4.41	-	2431	-	76.05%	-
		5	3.97	2.25	2473	1621	76.05%	76.05%
		10	5.35	3.26	2585	2087	76.18%	80.44%
	Gaussiano	1	9.27	2.22	2006	387	83.98%	83.44%
		5	18.26	4.46	1821	1042	84.39%	84.29%
		10	30.75	8.80	1794	1413	84.52%	84.36%

Tabla 6–16: Comparaciones entre Bajo Costo e Identificación de Variables con el grupo E

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**	
heart	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.04	0.02	85	20	85.18%	83.33%
N Datos		5	0.12	0.05	82	53	85.18%	83.33%
datos/test		10	0.16	0.11	82	70	85.18%	85.18%
216/54	Polinomial	1	0.01	0.009	155	50	83.33%	83.33%
		5	0.02	0.02	117	57	83.33%	83.33%
		10	0.02	0.02	110	60	81.48%	85.18%
	Gaussiano	1	0.01	0.01	116	38	83.33%	83.33%
		5	0.02	0.01	99	44	83.33%	83.33%
		10	0.04	0.02	100	69	85.18%	85.18%
ionosphere	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.1	0.05	78	52	85.71%	84.28%
		5	0.17	0.10	60	45	85.71%	87.14%
		10	0.21	0.11	57	37	85.71%	82.85%
281/70	Polinomial	1	0.02	0.09	205	91	64.28%	64.28%
		5	0.03	0.02	189	85	84.28%	84.28%
		10	0.04	0.03	168	81	85.71%	81.42%
	Gaussiano	1	0.04	0.02	116	72	92.85%	90%
		5	0.07	0.05	83	61	98.57%	94.28%
		10	0.07	0.05	74	60	98.57%	97.14%
liver-disorders	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.02	0.02	230	61	63.76%	56.52%
		5	0.07	0.04	215	85	66.66%	66.66%
		10	0.1	0.08	210	26	66.66%	65.21%
276/69	Polinomial	1	0.01	0.01	237	127	57.97%	43.47%
		5	0.01	0.02	233	77	59.42%	65.21%
		10	0.02	0.03	233	112	63.76%	66.66%
	Gaussiano	1	0.02	0.02	237	75	63.76%	59.42%
		5	0.04	0.03	222	66	68.11%	68.11%
		10	0.06	0.04	217	84	69.56%	66.66%
sonar	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.03	0.03	76	51	78.04%	73.17%
		5	0.08	0.05	66	52	75.60%	80.48%
		10	0.08	0.08	64	49	75.60%	78.04%
167/41	Polinomial	1	0.01	0.007	157	44	58.53%	60.97%
		5	0.01	0.009	138	47	65.85%	65.85%
		10	0.01	0.01	124	48	70.73%	63.41%
	Gaussiano	1	0.01	0.008	128	34	82.92%	65.85%
		5	0.02	0.02	102	44	78.04%	75.60%
		10	0.03	0.02	93	81	75.60%	75.60%

El análisis de estas tablas revela que para los problemas con número de datos de entrenamiento pequeños los tiempos tanto para el algoritmo de Bajo Costo y para el de Identificación de Variables se mantuvieron cercanos, mientras que para los problemas con tamaño mediano el algoritmo de Identificación de Variables se comporta muy bien, mejorando el tiempo de resolución con cualquier Kernel y parámetro C usado.

Tabla 6–17: Comparaciones entre Bajo Costo e Identificación de Variables con el grupo F

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**	
australian	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	1.53	0.07	178	16	84.78%	84.78%
Nro. Datos		5	5.27	1.03	180	174	84.78%	84.78%
entrenamiento/test		10	7.36	0.85	189	208	84.78%	84.78%
552/138	Polinomial	1	0.11	0.06	245	86	84.78%	84.78%
		5	0.24	0.06	210	53	89.13%	86.23%
		10	0.28	0.07	204	73	87.68%	86.23%
	Gaussiano	1	0.25	0.07	200	48	84.78%	84.78%
		5	0.39	0.10	193	77	89.13%	84.05%
		10	0.59	0.22	194	144	87.68%	86.23%
breast-cancer	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.07	0.05	47	22	97.79%	97.79%
		5	0.15	0.08	43	28	97.79%	97.79%
		10	0.17	0.08	44	16	97.79%	97.79%
547/136	Polinomial	1	0.05	0.04	80	43	98.52%	98.52%
		5	0.06	0.05	55	47	98.52%	98.52%
		10	0.07	0.04	58	27	98.52%	98.52%
	Gaussiano	1	0.06	0.04	63	32	98.52%	98.52%
		5	0.1	0.06	55	33	98.52%	98.52%
		10	0.12	0.08	61	44	97.79%	97.79%
diabetes	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.2	0.08	337	34	76.47%	76.47%
		5	0.54	0.19	326	150	76.47%	77.77%
		10	0.9	0.29	325	16	75.81%	75.81%
615/153	Polinomial	1	0.03	0.05	434	166	67.32%	69.28%
		5	0.08	0.06	396	162	77.12%	72.54%
		10	0.13	0.07	376	125	76.47%	73.85%
	Gaussiano	1	0.1	0.06	369	103	76.47%	74.50%
		5	0.23	0.1	328	133	75.81%	73.20%
		10	0.38	0.18	326	212	76.47%	75.16%
fourclass	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	0.18	0.07	392	49	84.88%	83.72%
		5	0.6	0.21	389	51	84.88%	83.13%
		10	1.04	0.31	386	150	84.88%	84.88%
690/172	Polinomial	1	0.17	0.15	407	119	80.23%	80.23%
		5	0.47	0.39	356	279	83.13%	85.46%
		10	0.5	0.33	348	214	85.46%	83.72%
	Gaussiano	1	0.28	0.07	353	54	86.62%	86.04%
		5	0.58	0.18	332	175	87.20%	83.13%
		10	0.87	0.44	320	42	90.11%	87.20%

Se puede decir que de todos los problemas resueltos usando identificación de variables 223 de ellos mejoraron el tiempo de ejecución, esto representa un 72.87% del total. Además, aunque el número de vectores de soporte de la columna Identificación sea menor o igual al número de vectores de soporte de la columna de Bajo Costo, se siguió manteniendo un error de generalización aceptable, en algunos mejoró el porcentaje de datos bien clasificados y solamente en el problema **W2a** se desmejoró mucho el porcentaje de datos bien clasificados para los Kernel Polinomial y Gaussiana con C = 10.

Tabla 6–18: Comparaciones entre Bajo Costo e Identificación de Variables con el grupo G

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**	
splice	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	2.02	0.48	413	80	84.78%	84.22%
Nro. Datos		5	6.79	0.91	412	152	84.68%	83.77%
entrenamiento/test		10	8.96	1.37	417	177	83.77%	84.09%
1000/2175	Polinomial	1	0.2	0.06	996	278	53.65%	57.01%
		5	0.33	0.18	960	587	86.06%	76.55%
		10	0.27	0.34	937	809	86.75%	84.32%
	Gaussiano	1	0.5	0.18	614	302	88.96%	86.29%
		5	0.67	0.42	576	492	90.11%	89.83%
		10	0.71	0.46	595	582	89.56%	89.70%
svmguide1	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	3.74	0.92	557	358	48.87%	48.77%
		5	7.48	1.46	444	278	49.05%	49.17%
		10	12.81	1.57	410	247	49.02%	49.27%
3089/4000	Polinomial	1	1.71	0.60	768	414	49.12%	49.12%
		5	3.19	0.74	504	391	49.57%	49.6%
		10	3.17	1.06	437	347	49.6%	49.67%
	Gaussiano	1	3.67	0.86	638	424	48.85%	48.77%
		5	6.47	1.01	435	319	49.3%	49.37%
		10	10.46	1.44	386	308	49.32%	49.47%
svmguide3	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	2.97	0.29	548	307	17.07%	51.21%
		5	7.77	1.07	534	58	39.02%	46.34%
		10	12.26	7.55	526	422	41.46%	48.78%
1243/41	Polinomial	1	0.56	0.44	594	249	0%	0%
		5	0.95	0.60	582	498	0%	0%
		10	1.32	0.91	576	518	2.43%	2.43%
	Gaussiano	1	1.12	0.19	573	378	2.43%	9.75%
		5	2.17	1.01	558	544	12.19%	17.07%
		10	2.63	2.28	537	528	19.51%	29.26%
german.numer	**kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	1.4	0.29	429	30	78.50%	79%
		5	3.85	1.73	428	272	77.5%	76%
		10	6.05	3.05	428	299	78%	76.5%
800/200	Polinomial	1	0.13	0.08	506	188	73%	73%
		5	0.27	0.15	475	212	77.50%	78%
		10	0.57	0.27	462	263	78%	77.5%
	Gaussiano	1	0.28	0.1	482	132	75%	75%
		5	0.45	0.19	461	222	79.50%	77.5%
		10	0.84	0.39	452	295	77.50%	77%

Tabla 6–19: Comparaciones entre Bajo Costo e Identificación de Variables con el grupo H

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**	
w1a	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	4.54	1.02	172	170	97.74%	97.71%
Nro. Datos		5	11.68	2.07	170	375	97.51%	97.35%
entrenamiento/test		10	16.77	2.33	386	387	97.46%	97.24%
2477/47272	Polinomial	1	1.02	1.64	2467	2086	97.02%	97.02%
		5	8.74	10.50	1277	1278	97.02%	97.02%
		10	12.16	0.15	1174	72	97.02%	96.86%
	Gaussiano	1	4.11	4.98	240	240	97.02%	97.02%
		5	4.58	0.80	236	225	97.25%	97.25%
		10	4.42	0.55	217	188	97.37%	97.35%
w2a	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	9.43	2.13	229	217	98.07%	98.08%
		5	15.52	3.42	207	200	97.80%	97.71%
		10	29.57	5.49	201	198	97.53%	97.39%
3470/46279	Polinomial	1	2.15	2.81	3058	2994	97.03%	97.03%
		5	16.1	8.2	1542	1492	97.03%	97.03%
		10	11.94	0.28	1708	108	97.03%	3.03%
	Gaussiano	1	6.7	3.53	291	292	97.03%	97.03%
		5	8.59	0.70	288	225	97.22%	70.70%
		10	8.97	1.18	295	259	97.59%	39.20%
w3a	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	22.33	5.61	281	282	98.29%	98.21%
		5	58.82	8.50	261	253	98.17%	98.14%
		10	93.45	11.84	253	270	98.01%	97.96%
4912/44837	Polinomial	1	5.96	7.72	3100	4907	98.01%	97.02%
		5	31.91	49.41	2134	1932	97.02%	97.02%
		10	67.09	0.15	1757	144	97.02%	96.60%
	Gaussiano	1	13.93	1.05	365	305	97.02%	97.11%
		5	14.53	1.95	352	334	97.31%	97.32%
		10	16.66	3.06	354	342	97.70%	97.73%
gisette	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	530.75	44.43	1080	991	97.9%	97.5%
		5	580.65	46.77	1080	982	97.9%	97.7%
		10	581.50	47	1080	975	97.9%	97.6%
6000/1000	Polinomial	1	519.60		1627		97.8%	
		5	506.28		1462		98%	
		10	517.64		1484		98%	
	Gaussiano	1	505.55		1666		97.7%	
		5	493.55		1363		98%	
		10	495.16		1400		97.9%	
mushrooms	**Kernel**	**C**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**	**Bajo Costo**	**Identificación**
	Lineal	1	39.57	8.48	400	339	100%	100%
		5	78.26	12.10	505	236	100%	100%
		10	68.79	14.40	480	217	100%	100%
6500/1624	Polinomial	1	36.77	47.35	3516	2227	93.71%	90.20%
		5	45.22	41.30	1819	1397	98.83%	95.75%
		10	47.64	37.86	1349	973	99.87%	98.15%
	Gaussiano	1	37.91		565		99.87%	
		5	60.05		292		99.83%	
		10	56.59		309		100%	

6.3.4 Comparaciones entre ASL y Bajo Costo con Identificación de Variable

Como parte de los objetivos de esta investigación, se procede a comparar el algoritmo de Bajo Costo usando Identificación de variables con ASL, ya que con éste algoritmo se obtuvo mejores resultados en el tiempo de corrida. En las siguientes tablas se estarán comparando el tiempo en segundos de ambos algoritmos, seguido con el número de vectores soporte y finalmente el porcentaje de datos bien clasificados solo para Bajo Costo con Identificación de variable.

Tabla 6–20: Comparaciones entre ASL e Identificación de Variable con el grupo A

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**
adwords1	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.004	0.02	50	48	100%
		5	0.01	0.006	50	16	100%
datos/dimension		10	0.01	0.16	50	48	100%
52/4702	Polinomial	1	0.008	0.003	52	18	58.33%
		5	0.008	0.004	52	38	50%
		10	0.004	0.005	52	32	50%
	Gaussiano	1	0.008	0.006	52	29	50%
		5	0.008	0.007	52	25	66.66%
		10	0.008	0.01	51	37	91.99%
adwords2	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.004	0.02	49	44	83.33%
		5	0.008	0.01	49	12	58.33%
		10	0.01	0.17	49	46	83.33%
52/3721	Polinomial	1	0.004	0.002	52	18	50%
		5	0.008	0.01	52	39	50%
		10	0.004	0.006	52	37	50%
	Gaussiano	1	0.008	0.01	52	28	50%
		5	0.008	0.01	52	32	75%
		10	0.008	0.009	51	38	83.33%
adwords3	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.001	0.007	47	42	83.33%
		5	0.001	0.01	47	46	66.66%
		10	0.001	0.03	47	47	66.66%
52/242	Polinomial	1	0.001	0.004	52	18	50%
		5	0.001	0.007	52	37	50%
		10	0.001	0.004	52	37	50%
	Gaussiano	1	0.001	0.004	51	27	50%
		5	0.001	0.005	51	22	50%
		10	0.001	0.01	51	33	50%
adwords4	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.001	0.005	47	37	100%
		5	0.001	0.01	48	48	100%
		10	0.001	0.01	48	47	100%
52/229	Polinomial	1	0.001	0.003	42	18	50%
		5	0.001	0.004	52	38	50%
		10	0.001	0.008	52	37	50%
	Gaussiano	1	0.001	0.005	52	23	50%
		5	0.001	0.003	52	26	83.33%
		10	0.001	0.009	49	29	83.33%

Al comparar el algoritmo de Bajo Costo que usa Identificación de Variables con ASL se puede observar que en la mayoría de los problemas (A-H) que involucran tanto problemas con alta dimensionalidad como número de datos de entrenamiento grandes y pequeños, que los problemas resueltos con los Kernel Polinomial y Gaussiano presentaron mejores resultados en tiempo que usando el Kernel Lineal.

Tabla 6–21: Comparaciones entre ASL e Identificación de Variable con el grupo B

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**
arsene	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.22	0.39	75	65	78.94%
		5	0.23	0.20	75	66	78.94%
datos/dimension		10	0.21	0.14	75	54	78.94%
80/10000	Polinomial	1	0.18	0.02	76	16	73.68%
		5	0.18	0.11	77	30	57.89%
		10	0.18	0.19	78	68	78.94%
	Gaussiano	1	0.20	0.04	80	18	63.15%
		5	0.20	0.15	80	60	68.42%
		10	0.20	0.21	80	77	84.21%
duke	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.02	0.01	27	4	57.14%
		5	0.02	0.01	27	4	57.14%
		10	0.02	0.01	27	4	85.71%
29/7129	Polinomial	1	0.01	0.006	29	7	42.85%
		5	0.02	0.02	29	19	42.85%
		10	0.02	0.03	27	22	57.14%
	Gaussiano	1	0.02	0.003	29	11	57.14%
		5	0.02	0.02	29	29	85.71%
		10	0.02	0.01	29	29	85.71%
leukemia	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.02	0.01	29	6	73.52%
		5	0.02	0.01	29	6	76.47%
		10	0.02	0.02	29	6	85.29%
38/7129	Polinomial	1	0.02	0.01	35	22	58.82%
		5	0.02	0.01	36	24	58.82%
		10	0.02	0.02	37	29	58.82%
	Gaussiano	1	0.03	0.02	38	29	58.82%
		5	0.02	0.03	38	38	70.58%
		10	0.03	0.03	38	38	70.58%
colon-cancer	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.008	0.01	26	4	50%
		5	0.01	0.02	26	4	50%
		10	0.01	0.02	26	4	50%
40/2000	Polinomial	1	0.008	0.002	39	20	60%
		5	0.008	0.01	38	36	60%
		10	0.01	0.01	38	38	60%
	Gaussiano	1	0.01	0.005	39	27	60%
		5	0.008	0.01	38	37	80%
		10	0.01	0.009	38	36	80%

Tabla 6–22: Comparaciones entre ASL e Identificación de Variable con el grupo C

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**
Farm-ads	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	-	61.71	-	1346	87.19%
		5	-	maxIter	-	maxIter	maxIter
datos/dimension		10	-	135.25	-	1251	87.19%
3315/54877	Polinomial	1	16.89	2.42	3315	958	52.53%
		5	16.90	9.67	3315	2227	53.26%
		10	16.93	12.94	3315	2261	53.26%
	Gaussiano	1	18.54	2.11	3117	735	53.26%
		5	18.70	8.98	2824	1059	75.84%
		10	18.83	7.93	2520	1021	75.48%
news20	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	-		-		
		5	-		-		
		10	-		-		
15997/1355191	Polinomial	1	561.16		15997		
		5	526.15		15997		
		10	565.71		15997		
	Gaussiano	1	798.52		15997		
		5	823.69		15996		
		10	844.05		15996		
rcv1	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	-		-		
		5	-		-		
		10	-		-		
20242/47236	Polinomial	1	279.15		20242		
		5	265.20		20242		
		10	267.83		20242		
	Gaussiano	1	336.56		20150		
		5	339.77		19620		
		10	376.11		19582		
dorothea	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.74	1.38	225	210	85.62%
		5	0.70	1.45	225	220	85.62%
		10	0.70	1.85	225	218	85.62%
641/937	Polinomial	1	1.15	1.12	410	318	90%
		5	1.10	0.25	405	251	90%
		10	1.18	1.21	405	391	90.62%
	Gaussiano	1	1.14	1.17	282	216	90%
		5	1.35	0.48	294	226	80.62%
		10	1.36	0.57	301	155	90.62%
dexter	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.1	0.04	237	30	53.33%
		5	0.1	0.06	237	36	68.33%
		10	0.1	0.09	237	50	65%
240/19999	Polinomial	1	1.3	0.09	240	58	61.66%
		5	1.33	0.01	240	20	63.33%
		10	1.3	0.01	240	24	61.66%
	Gaussiano	1	1.52	0.09	240	62	61.66%
		5	1.35	0.01	240	28	61.66%
		10	1.4	0.03	240	38	61.66%

Es importante destacar que en ninguna de las tablas anteriores y en las siguientes se hace referencia del porcentaje de datos bien clasificados para el algoritmo ASL ya que no estamos interesados en conocer los parámetros óptimos para hacer una buena clasificación sino cual es el comportamiento del método propuesto usando identificación para problemas reales tipo MVS.

Tabla 6–23: Comparaciones entre ASL e Identificación de Variable con el grupo D

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**
a1a	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.2	0.57	589	186	83.91%
		5	0.8	2.56	574	377	83.29%
datos/dimension		10	2.4	3.58	567	338	83.00%
1605 /123	Polinomial	1	0.3	-	831	-	-
		5	0.3	0.5	830	660	75.94%
		10	0.3	0.5	830	742	75.94%
	Gaussiano	1	0.5	0.27	754	287	83.55%
		5	0.6	0.43	660	173	84.02%
		10	0.7	0.46	645	189	83.70%
a2a	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.4	1.11	879	263	84.44%
		5	1.9	8.11	862	541	83.69%
		10	4.05	13.32	863	578	83.28%
2265/123	Polinomial	1	0.6	-	1196	-	-
		5	0.6	0.67	1192	963	76.00%
		10	0.7	0.75	1192	858	76.00%
	Gaussiano	1	1.1	0.48	1068	297	83.78%
		5	1.3	0.83	955	183	84.32%
		10	1.4	0.91	935	182	84.34%
a3a	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.8	3.54	1163	518	84.33%
		5	3.1	16.54	1148	707	83.81%
		10	5.7	36.21	1146	760	83.79%
3185/123	Polinomial	1	1.2	-	1591	-	-
		5	1.3	0.97	1591	1422	75.93%
		10	1.5	1.39	1591	1348	75.93%
	Gaussiano	1	2.3	1.06	1389	377	83.53%
		5	2.6	1.62	1249	210	84.22%
		10	2.9	3.86	1228	1031	84.43%
a4a	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	1.5	7.99	1736	665	84.56%
		5	7.5	58.09	1723	1199	84.42%
		10	12.56	73.81	1715	1151	84.26%
4781/123	Polinomial	1	2.8	-	2434	-	-
		5	2.9	2.25	2433	1621	76.05%
		10	3.6	3.26	2433	2087	80.44%
	Gaussiano	1	5.3	2.22	2002	387	83.44%
		5	6.2	4.46	1821	1042	84.29%
		10	7.9	8.80	1793	1413	84.36%

Tabla 6–24: Comparaciones entre ASL e Identificación de Variable con el grupo E

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**
heart	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.01	0.02	85	20	83.33%
		5	0.04	0.05	82	53	83.33%
datos/dimension		10	0.07	0.11	82	70	85.18%
216/13	Polinomial	1	0.004	0.009	155	50	83.33%
		5	0.008	0.02	117	57	83.33%
		10	0.1	0.02	110	60	85.18%
	Gaussiano	1	0.008	0.01	117	38	83.33%
		5	0.02	0.01	102	44	83.33%
		10	0.03	0.02	95	69	85.18%
ionosphere	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.03	0.05	78	52	84.28%
		5	0.06	0.10	60	45	87.14%
		10	0.08	0.11	57	37	82.85%
281/34	Polinomial	1	0.01	0.09	205	91	64.28%
		5	0.02	0.02	188	85	84.28%
		10	0.1	0.03	168	81	81.42%
	Gaussiano	1	0.02	0.02	120	72	90%
		5	0.03	0.05	79	61	94.28%
		10	0.04	0.05	69	60	97.14%
liver-disorders	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.01	0.02	230	61	56.52%
		5	0.02	0.04	215	85	66.66%
		10	0.02	0.08	210	26	65.21%
276/6	Polinomial	1	0.008	0.01	236	127	43.47%
		5	0.004	0.02	236	77	65.21%
		10	0.01	0.03	233	112	66.66%
	Gaussiano	1	0.02	0.02	237	75	59.42%
		5	0.008	0.03	217	66	68.11%
		10	0.03	0.04	209	84	66.66%
sonar	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.01	0.03	76	51	73.17%
		5	0.02	0.05	66	52	80.48%
		10	0.05	0.08	64	49	78.04%
167/60	Polinomial	1	0.004	0.007	157	44	60.97%
		5	0.008	0.009	138	47	65.85%
		10	0.008	0.01	124	48	63.41%
	Gaussiano	1	0.008	0.008	128	34	65.85%
		5	0.01	0.02	102	44	75.60%
		10	0.01	0.02	93	81	75.60%

Tabla 6–25: Comparaciones entre ASL e Identificación de Variable con el grupo F

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**
australian	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.4	0.07	165	16	84.78%
		5	1.7	1.03	162	174	84.78%
datos/dimension		10	3.1	0.85	162	208	84.78%
552/14	Polinomial	1	0.04	0.06	241	86	84.78%
		5	0.1	0.06	212	53	86.23%
		10	0.1	0.07	200	73	86.23%
	Gaussiano	1	0.08	0.07	208	48	84.78%
		5	0.1	0.10	195	77	84.05%
		10	0.1	0.22	184	144	86.23%
breast-cancer	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.02	0.05	47	22	97.79%
		5	0.08	0.08	43	28	97.79%
		10	0.1	0.08	43	16	97.79%
547/10	Polinomial	1	0.02	0.04	80	43	98.52%
		5	0.04	0.05	55	47	98.52%
		10	0.1	0.04	61	27	98.52%
	Gaussiano	1	0.04	0.04	63	32	98.52%
		5	0.06	0.06	55	33	98.52%
		10	0.07	0.08	61	44	97.79%
diabetes	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.04	0.08	336	34	76.47%
		5	0.2	0.19	324	150	77.77%
		10	0.2	0.29	324	16	75.81%
615/8	Polinomial	1	0.02	0.05	435	166	69.28%
		5	0.03	0.06	396	162	72.54%
		10	0.03	0.07	373	125	73.85%
	Gaussiano	1	0.06	0.06	367	103	74.50%
		5	0.08	0.1	328	133	73.20%
		10	0.1	0.18	324	212	75.16%
fourclass	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.03	0.07	389	49	83.72%
		5	0.1	0.21	386	51	83.13%
		10	0.02	0.31	385	150	84.88%
690/2	Polinomial	1	0.04	0.15	407	119	80.23%
		5	0.08	0.39	355	279	85.46%
		10	0.1	0.33	344	214	83.72%
	Gaussiano	1	0.1	0.07	349	54	86.04%
		5	0.1	0.18	326	175	83.13%
		10	0.2	0.44	318	42	87.20%

Tabla 6–26: Comparaciones entre ASL e Identificación de Variable con el grupo G

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**
splice	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.5	0.48	404	80	84.22%
		5	3	0.91	408	152	83.77%
datos/dimension		10	4.4	1.37	407	177	84.09%
1000/60	Polinomial	1	0.1	0.06	996	278	57.01%
		5	0.2	0.18	960	587	76.55%
		10	0.3	0.34	937	809	84.32%
	Gaussiano	1	0.3	0.18	607	302	86.29%
		5	0.4	0.42	569	492	89.83%
		10	0.47	0.46	587	582	89.70%
svmguide1	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.2	0.92	554	358	48.77%
		5	0.3	1.46	434	278	49.17%
		10	0.5	1.57	409	247	49.27%
3089/4	Polinomial	1	0.4	0.60	767	414	49.12%
		5	0.6	0.74	504	391	49.6%
		10	0.8	1.06	437	347	49.67%
	Gaussiano	1	1.3	0.86	630	424	48.77%
		5	1.5	1.01	434	319	49.37%
		10	1.8	1.44	386	308	49.47%
svmguide3	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.1	0.29	546	307	51.21%
		5	0.5	1.07	529	58	46.34%
		10	0.9	7.55	521	422	48.78%
1243/21	Polinomial	1	0.1	0.44	589	249	0%
		5	0.2	0.60	579	498	0%
		10	0.3	0.91	576	518	2.43%
	Gaussiano	1	0.3	0.19	572	378	9.75%
		5	0.4	1.01	546	544	17.07%
		10	0.4	2.28	515	528	29.26%
german.numer	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.2	0.29	428	30	79%
		5	0.6	1.73	426	272	76%
		10	0.7	3.05	426	299	76.5%
800/24	Polinomial	1	0.07	0.08	506	188	73%
		5	0.1	0.15	475	212	78%
		10	0.2	0.27	462	263	77.5%
	Gaussiano	1	0.1	0.1	505	132	75%
		5	0.3	0.19	478	222	77.5%
		10	0.3	0.39	457	295	77%

Tabla 6–27: Comparaciones entre ASL e Identificación de Variable con el grupo H

Problema			**Tiempo seg**		N_{sv}		**Error de Generalización**
w1a	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.2	1.02	172	170	97.71%
		5	0.5	2.07	161	375	97.35%
datos/dimension		10	0.5	2.33	400	387	97.24%
2477/300	Polinomial	1	0.7	1.64	2091	2086	97.02%
		5	0.8	10.50	1891	1278	97.02%
		10	0.9	0.15	1974	72	96.86%
	Gaussiano	1	1.5	4.98	246	240	97.02%
		5	1.7	0.80	237	225	97.25%
		10	1.5	0.55	215	188	97.35%
w2a	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.3	2.13	230	217	98.08%
		5	0.6	3.42	207	200	97.71%
		10	1.3	5.49	200	198	97.39%
3470/300	Polinomial	1	1.3	2.81	2809	2994	97.03%
		5	1.7	8.2	2389	1492	97.03%
		10	2	0.28	2170	108	3.03%
	Gaussiano	1	2.8	3.53	295	292	97.03%
		5	2.8	0.70	286	225	70.70%
		10	3.1	1.18	292	259	39.20%
w3a	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.6	5.61	279	282	98.21%
		5	1.5	8.50	259	253	98.14%
		10	4	11.84	251	270	97.96%
4912/300	Polinomial	1	2.6	7.72	4215	4907	97.02%
		5	4.2	49.41	2850	1932	97.02%
		10	4.7	0.15	2718	144	96.60%
	Gaussiano	1	5.6	1.05	367	305	97.11%
		5	5.9	1.95	354	334	97.32%
		10	6.2	3.06	351	342	97.73%
gisette	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	16.25	44.43	1084	991	97.5%
		5	17.95	46.77	1084	982	97.7%
		10	17.25	47	1084	975	97.6%
6000/5000	Polinomial	1	543.36		1631		
		5	556.96		1464		
		10	556.32		1487		
	Gaussiano	1	573.80		1670		
		5	577.66		1370		
		10	584.09		1398		
mushrooms	**Kernel**	**C**	**ASL**	**Identificación**	**ASL**	**Identificación**	**Identificación**
	Lineal	1	0.73	8.48	464	339	100%
		5	0.76	12.10	415	236	100%
		10	0.71	14.40	426	217	100%
6500/112	Polinomial	1	7.74	47.35	3647	2227	90.20%
		5	10.31	41.30	1809	1397	95.75%
		10	10.98	37.86	1343	973	98.15%
	Gaussiano	1	14.41		592		
		5	15.48		318		
		10	15.94		301		

CAPÍTULO VII
CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se ha propuesto un algoritmo de bajo costo basado en los métodos de punto interior que resuelven los problemas cuadráticos y convexos que surgen de las Máquinas de Vectores de Soporte de manera práctica, sencilla, con bajo costo computacional y en rápido tiempo, por medio de la longitud espectral.

Las pruebas realizadas para problemas SVM artificiales, demostraron que el método propuesto comparado con el método Predictor-Corrector Mehrotra programado en este trabajo, resuelven rápidamente con Kernel Lineal los problemas donde los datos de entrenamiento son linealmente separables sin importar el valor del párametro C y el rango de los datos de entrenamiento.

Para los problemas artificiales generados con datos linealmente no separables y no lineales en el rango [0,1], el algoritmo de bajo costo sigue siendo competitivo, pero su complejidad aumenta cuando el parámetro C aumenta. El aporte de éste método en esta invetigación es que se pueden resolver problemas de mayor tamaño sin afectar la memoria del computador. Una característica especial que presenta el algoritmo en corrida es que las medidas relativas de la infactibilidad primal tienden rápidamente a cero mientras que la medida relativa de infactibilidad dual es la que más tarde en converger a cero ya que a medida que se acerca a la solución ésta comienza a decrecer lentamente.

Reprogramando el código de MATLAB a lenguaje C++ usando las funciones programadas y optimizadas de LIBSVM, los problemas reales resueltos por el algoritmo de bajo costo sin paso predictor fueron competitivos en tiempo con LIBSVM y ASL para aquellos problemas con datos de entrenamiento pequeños o con alta dimensionalidad. Se observa también que el algoritmo de bajo costo sin paso predictor se comporta mejor cuando el Kernel es Polinomial o Gaussiano.

La razón de este comportamiento se desconoce, pero que cree que esta relacionada con el mal condicionamiento de la matriz Hessiana, y como la dirección de búsqueda usa la longitud espectral ésta pareciera perder información. Si la dirección de búsqueda no avanza a la solución la longitud de paso es muy pequeña y esto hace que aumente el número de iteraciones y el tiempo de corrida del método propuesto.

En esta investigación surge otro método de bajo costo que llamamos identificación de variables y se combina con el algoritmo de bajo costo sin paso predictor. Los resultados del método propuesto mejoraron cuando se resolvieron los problemas con un subconjunto de datos de entrenamiento que contienen aquellas variables candidatas a ser vectores de soporte no acotado. La identificación hace que se reduzca el número de datos de entrenamiento y se eliminen datos que no aportan información al modelo o sean muy mal clasificados.

Las comparaciones realizadas con porcentaje de datos bien clasificados entre el algoritmo de bajo costo sin paso predictor y bajo costo con identificación de variables, demostraron que a pesar de resolverse un subconjunto de datos candidatos a ser vectores de soporte no acotados se sigue manteniendo en la gran mayoría de los resultados el mismo o mejor porcentaje de datos bien clasificados.

También se puede decir que el algoritmo de bajo costo con identificación de variables copara favorablemente con los tiempos de ASL, pero la única desventaja de este algoritmo, es que una parte está programado en MATLAB lo que nos limita

a resolver sólo aquellos problemas que tienen datos de entrenamiento por debajo de 6000 muestras.

Como conclusiones finales, se ha obtenido avances significativos en lo que respecta a resolver problemas de tipo SVM con el algoritmo de bajo costo propuesto en esta investigación. Sin embargo, resolver problemas reales de manera eficiente, no sólo se necesita la parte teórica del algoritmo sino una buena estrategia computacional para administrar de forma eficiente la memoria del computador y disminuir el tiempo de corrida, así que las recomendaciones para trabajos futuros derivadas del siguiente trabajo de tesis se muestran a continuación:

- Resolver los problemas reales con el algoritmo de bajo costo usando descomposición.
- Programar el algoritmo de bajo costo con identificación de variables en lenguaje $C++$ con el fin de resolver problemas con más datos de entrenamiento.
- Describir teóricamente el comportamiento de las direcciones en el paso predictor al usar la longitud espectral.
- Usar otras técnicas de programación avanzada tales como programación en paralelo.
- Desarrollar teoría de convergencia.
- Generalizar el algoritmo de bajo costo para resolver problemas con más de dos clases.

APÉNDICES

APÉNDICE A

CODIGO DEL PROGRAMA EN MATLAB

Bajo_costo_predictor_corrector_b.m

```
%=========================================================================
function [x_stop,iter,fobj]=bajo_costo_predictor_corrector_b(ns,x,s
    ,t,u,w,y,H,b,e,C)
%=========================================================================
% La funcion implementa el metodo de punto interior
% primal-dual modificado de puntos interiores para programacion
    cuadratica.
%=========================================================================
% ENTRADAS:
% ns: dimension del problema.
% x: punto inicial primal.
% s: punto inicial dual.
% t: punto inicial dual.
% u: punto inicial dual.
% w: punto inicial primal.
% y: vector de restricciones.
% H: matriz asociada a la funcion objetivo.
% b: vector asociado a las restricciones.
% e: vector asociado a la funcion objetivo.
% C: vector de cotas superiores.
%=========================================================================
% SALIDAS:
% x_stop: punto optimo.
% iter: numero de iteraciones.
% fobj: Valor en la funcion objetivo
```

```
%=========================================================================

format long
counter_max = 100000;
sigfig_max = 6.5;
proceso = 1;
tol = 1.0e-05;  % condicion de parada
eta = 0.95;     % factor para fatibilidad extricta
tol1 = 1.0e-05;

% Calculo de los residuales
[n,m]=size(y);
g0 = H*x-e;
re = g0 - s*y - t + u;
rb=y'*x-b;
rc=C-w-x;
rtx = t.*x;
ruw = u.*w;

sigma = 0.001;  % parametro de centrado
mu = (t'*x + u'*w)/(2*n); % medida de complementaria

iter=0;

nrb = norm(rb,inf)/(1+norm(b,inf));
nre = norm(re,inf)/(1+norm(e,inf));
nrc = norm(rc,inf)/(1+norm(C,inf));

absmu = 1;

fprintf('\n            Objetivo            Objetivo
   Cifras         Longitud        Medida de           Error
   Error     Error');
fprintf('\n             primal               Dual
   Significativas      de paso     complementariedad     relativo
     relativo  relativo');
```

```
fprintf('\n iter   0.5xt*H*x - et*x   -0.5xt*H*x - Ct*u    max(-
    log10(*),0)    alpha_p            mu               ||re||     ||
    rb||    ||rc||\n');

lambda=max(10^(-5),norm(g0,'inf')); % lambda_0

while (nrb > tol || nre > tol || nrc > tol || absmu > tol)

    % Resolver el sistema con Newton.

    V1 = -re-rtx./x+(ruw+u.*rc)./w;
    V2 = -rb;

    D = lambda + t./x + u./w;

    K1 = y.*V1./D;
    K2 = y.*y./D;

    % paso predictor
    ds = -(sum(K1)-V2)/sum(K2);
    dx = (y*ds + V1)./D;
    dw = -dx + rc;
    dt = (-rtx-t.*dx)./x;
    du = (-ruw-u.*dw)./w;

    d = 1;
    % Calcular alpha_aff
    alphax=(1/max(1,max(-dx./(d*x))));
    alphaw=(1/max(1,max(-dw./(d*w))));
    alphat=(1/max(1,max(-dt./(d*t))));
    alphau=(1/max(1,max(-du./(d*u))));

    alpha_a = min(min(alphax,alphaw),min(alphat,alphau));

    %Compute the affine duality gap
```

```
mu_a = ((x+alpha_a*dx)'*(t+alpha_a*dt)+(w+alpha_a*dw)'*(u+
    alpha_a*du))/(2*n);

%Compute the centering parameter
sigma = (mu_a/mu)^3;

if sigma>1
    sigma=0.9
end

tau = sigma*mu;

%Solve system
if (nrb > tol1 && nrc > tol1)
rtx = rtx - tau*e + dx.*dt;
ruw = ruw - tau*e + dw.*du;
else
rtx = rtx - tau*e;
ruw = ruw - tau*e;
end

V1 = -re-rtx./x+(ruw+u.*rc)./w;
V2 = -rb;

K1 = y.*V1./D;
K2 = y.*y./D;

ds = -(sum(K1)-V2)/sum(K2);
dx = (y*ds + V1)./D;
dw = -dx + rc;
dt = (-rtx-t.*dx)./x;
du = (-ruw-u.*dw)./w;

d = 1;
% Calcular alpha
alphax=(1/max(1,max(-dx./(d*x))));
```

```
alphaw=(1/max(1,max(-dw./(d*w))));
alphat=(1/max(1,max(-dt./(d*t))));
alphau=(1/max(1,max(-du./(d*u))));

alpha_p = eta*min(alphax,alphaw);
alpha_d = eta*min(alphat,alphau);

% Actualizar x,s,t,u,w

xn = x+alpha_p*dx;
s  = s+alpha_d*ds;
t  = t+alpha_d*dt;
u  = u+alpha_d*du;
w  = w+alpha_p*dw;

iter=iter+1;

% Actualizar los residuales
g = H*xn - e;
re = g - s*y - t + u;
rb=y'*xn-b;
rc=C-w-xn;
rtx = t.*xn;
ruw = u.*w;

mu = (t'*xn + u'*w)/(2*n);

lambda=lspect(xn,x,g,g0);

g0=g;
x=xn;

nrb = norm(rb,inf)/(1+norm(b,inf));
nre = norm(re,inf)/(1+norm(e,inf));
nrc = norm(rc,inf)/(1+norm(C,inf));
absmu = abs(mu);
```

```
    x_h_x = x'*(g0+e);

    primal_obj = 0.5 * x_h_x - e'*x;
    dual_obj = -0.5 * x_h_x - C'*u;

    sigfig = log10(abs(primal_obj) + 1) - log10(abs(primal_obj -
        dual_obj));
    sigfig = max(sigfig, 0);

    if (iter > counter_max), status = 'ITERATION_LIMIT', break; end
    if (sigfig  > sigfig_max),  status = 'OPTIMAL_SOLUTION', break;
        end

%fprintf('%4i | %13.2e | % 16.2e | %12.3f | %17.4f | %12.2e | %17.2
    e | %.2e | %.2e\n', iter, primal_obj, dual_obj, sigfig, alpha_p,
    mu, nre, nrb, nrc);

end
fprintf('%4i | %13.2e | % 16.2e | %12.3f | %17.4f | %12.2e | %17.2e
    | %.2e | %.2e\n', iter, primal_obj, dual_obj, sigfig, alpha_p,
   mu, nre, nrb, nrc);
fobj = 0.5*x'*H*x - sum(x);
%Salida
x_stop = x;

if proceso == 1
    fprintf('\n Optimizacion terminada con exito\n');
end
```

buildQ.m

```
function [Q]=buildQ(a,X,func)
% Este codigo construye la matriz Q de acuerdo al tipo de kernel
    usado.
%-------------------------------------------------------------
%       Q: Q(i,j)=a_i a_j K(X_i,X_j): los kernels considerados son:
```

```
%       func:
%           0: no kernel (K(X,Y)=X^t Y)
%           1: K(X,Y)=(coef+gamma*X^tY)^2
%           2: K(X,Y)=exp(-gamma^(-1)||X-Y||^2)
%           3: K(X,Y)=(1+X^tY)^5
%


coef = 0;


[dim,ns]=size(X);
if (func==0)
    K = X'*X;
    Q =  K.*(a'*a);
elseif (func==3)
    ps = X'*X;
    K = (1/dim*ps + coef).^5;
    Q =  K.*(a'*a);
elseif (func==1)
    ps = X'*X;
    K = (1/dim*ps + coef).^3;
    Q =  K.*(a'*a);
elseif (func == 2)


    ps = X'*X;
    [nps,pps]=size(ps);
    normx = sum(X.^2,1);
    normxsup = sum(X.^2,1);
    ps = -2*ps + repmat(normx,pps,1) + repmat(normxsup',1,nps) ;
    K = exp(-1/dim*ps);
    Q =  K.*(a'*a);


end
end
```

lspect.m

```
% Esta funcion  obtiene la longitud espectral.
```

```
function lambda=lspect(x,xant,g,gant)
lmin=1.e-5;
lmax=1.e+5;
sant=x-xant;
yant=g-gant;
sts=sant'*sant;
sty=sant'*yant;
lambda= max(lmin,sty/sts);
```

MPI_tradicional.m

```
%=========================================================================
function [x_stop,iter,fobj]=MPI_tradicional(ns,x,s,t,u,w,y,H,b,e,C)
%=========================================================================
% La funcion  implementa el metodo de punto interior
% primal-dual tradicional de puntos interiores para programacion
    cuadratica.
%=========================================================================
% ENTRADAS:
% ns: dimension del problema.
% x: punto inicial primal.
% s: punto inicial dual.
% t: punto inicial dual.
% u: punto inicial dual.
% w: punto inicial primal.
% y: vector de restricciones.
% H: matriz asociada a la funcion objetivo.
% b: vector asociado a las restricciones.
% e: vector asociado a la funcion objetivo.
% C: vector de cotas superiores.
%=========================================================================
% SALIDAS:
% x_stop: punto optimo.
% iter: numero de iteraciones.
% fobj: Valor en la funcion objetivo
%=========================================================================


format long
```

```
tol = 1.0e-05;  % condicion de parada
eta = 0.95;     % factor para fatibilidad extricta

% Calculo de los residuales
[n,m]=size(y);
g0 = H*x-e;
re = g0 - s*y - t + u;
rb=y'*x-b;
rc=C-w-x;
rtx = t.*x;
ruw = u.*w;

sigma = 0.001;  % parametro de centrado
mu = (t'*x + u'*w)/(2*n); % medida de complementaria
mu_1 = (t'*x)/n;
mu_2 = (u'*w)/n;

iter=0;

nrb = norm(rb,inf)/(1+norm(b,inf));
nre = norm(re,inf)/(1+norm(e,inf));
nrc = norm(rc,inf)/(1+norm(C,inf));

bc = 1+max(norm(b,inf),max(norm(e,inf),norm(C,inf)));
residuo = norm([re;rb;rc;rtx;ruw],inf)/bc;
nrCC = norm([rb;rc],inf);

absmu = 1;

fprintf('\n                No Fac.            Error       Brecha
           Brecha     Error');
fprintf('\n                  primal           Lagrange       dual
             dual     relativo');
fprintf('\n iter    y"x -b, C*e -x -w   G*x-e -s*y-t+u    t"x
              u"w       TOTAL\n');
```

```
fprintf('%5i %15.2e %17.2e %13.2e %13.2e %10.2e\n',iter,nrCC,norm(
    re),mu_1,mu_2,residuo);

while (nrb > tol || nre > tol || nrc > tol || absmu > tol)

    % Resolver el sistema con Newton.
        % matriz para resolver el sistema
    M=[H+sparse(diag(t./x+u./w)) -y; -y' 0];

    % paso predictor

    [L,D,P] = ldl(M);
    vecd=[-re-rtx./x+ruw./w+u.*rc./w;rb];
    vsol = P*(L'\(D\(L\(P'*vecd))));
    dx=vsol(1:n);
    ds=vsol(n+1);
    dw = -dx + rc;
    dt = (-rtx-t.*dx)./x;
    du = (-ruw-u.*dw)./w;

    % Calcular alfa_aff
    alpha_a=1;
    idx_x = find(dx < 0);
    if(isempty(idx_x)==0)
        alpha_a=min(alpha_a,min(-x(idx_x)./dx(idx_x)));
    end
    idx_w = find(dw < 0);
    if(isempty(idx_w)==0)
        alpha_a=min(alpha_a,min(-w(idx_w)./dw(idx_w)));
    end
    idx_t = find(dt < 0);
    if(isempty(idx_t)==0)
        alpha_a=min(alpha_a,min(-t(idx_t)./dt(idx_t)));
    end
    idx_u = find(du < 0);
    if(isempty(idx_u)==0)
```

```
    alpha_a=min(alpha_a,min(-u(idx_u)./du(idx_u)));
end

%Calcular el gap dual afin
mu_a = ((x+alpha_a*dx)'*(t+alpha_a*dt)+(w+alpha_a*dw)'*(u+
    alpha_a*du))/(2*n);

%Calcular el parametro de centrado
sigma = (mu_a/mu)^3;

tau = sigma*mu;

%Resolver el sistema
rtx = rtx - tau*e + dx.*dt;
ruw = ruw - tau*e + dw.*du;

% paso corrector
vecd=[-re-rtx./x+ruw./w+u.*rc./w;rb];
vsol = P*(L'\(D\(L\(P'*vecd))));
dx=vsol(1:n);
ds=vsol(n+1);
dw = rc - dx;
dt = (-rtx-t.*dx)./x;
du = (-ruw-u.*dw)./w;

% Calcular alfa
alphax=1;
idx_x = find(dx < 0);
if(isempty(idx_x)==0)
    alphax=min(1,min(-x(idx_x)./dx(idx_x)));
end
alphaw=1;
idx_w = find(dw < 0);
if(isempty(idx_w)==0)
    alphaw=min(1,min(-w(idx_w)./dw(idx_w)));
end
```

```
alphat=1;
idx_t = find(dt < 0);
if(isempty(idx_t)==0)
    alphat=min(1,min(-t(idx_t)./dt(idx_t)));
end
alphau=1;
idx_u = find(du < 0);
if(isempty(idx_u)==0)
    alphau=min(1,min(-u(idx_u)./du(idx_u)));
end

alpha_p = eta*min(alphax,alphaw);
alpha_d = eta*min(alphat,alphau);

alpha = min(alpha_p,alpha_d);

% Actualizar s,t,u,w
x = x+alpha_p*dx;
s = s+alpha_d*ds;
t = t+alpha_d*dt;
u = u+alpha_d*du;
w = w+alpha_p*dw;
iter=iter+1;

% Actualizar los residuales
g = H*x - e;
re = g - s*y - t + u;
rb=y'*x-b;
rc=C-w-x;
rtx = t.*x;
ruw = u.*w;

mu = (t'*x + u'*w)/(2*n);
mu_1 = (t'*x)/n;
mu_2 = (u'*w)/n;
```

```
    nrb = norm(rb,inf)/(1+norm(b,inf));
    nre = norm(re,inf)/(1+norm(e,inf));
    nrc = norm(rc,inf)/(1+norm(C,inf));

    absmu = abs(mu);

    bc = 1+max(norm(b,inf),max(norm(e,inf),norm(C,inf)));
    residuo = norm([re;rb;rc;rtx;ruw],inf)/bc;
    nrCC = norm([rb;rc],inf);

%fprintf('%5i %15.2e %17.2e %13.2e %13.2e %10.2e\n',iter,nrCC,norm(
    re),mu_1,mu_2,residuo);

end
fprintf('%5i %15.2e %17.2e %13.2e %13.2e %10.2e\n',iter,nrCC,norm(
    re),mu_1,mu_2,residuo);
fobj = 0.5*x'*H*x - sum(x);
%Salida
x_stop = x;
```

bajo_costo_sin_predictor.m

```
%=========================================================================
function [x_stop,iter,fobj,SV]=bajo_costo_sin_predictor(ns,x,s,t,u,
    w,y,H,b,e,C)
%=========================================================================
% La funcion implementa el metodo de punto interior
% primal-dual modificado de puntos interiores para programacion
    cuadratica.
%=========================================================================
% ENTRADAS:
% ns: dimension del problema.
% x: punto inicial primal.
% s: punto inicial dual.
% t: punto inicial dual.
% u: punto inicial dual.
% w: punto inicial primal.
```

```
% y: vector de restricciones.
% H: matriz asociada a la funcion objetivo.
% b: vector asociado a las restricciones.
% e: vector asociado a la funcion objetivo.
% C: vector de cotas superiores.
%=========================================================
% SALIDAS:
% x_stop: punto optimo.
% iter: numero de iteraciones.
% fobj: Valor en la funcion objetivo
%=========================================================

format long
counter_max = 100000;
sigfig_max = 6.5;
proceso = 1;
tol = 1.0e-05;  % condicion de parada
eta = 0.95;     % factor para fatibilidad extricta

% Calculo de los residuales
[n,m]=size(y);
g0 = H*x-e;
re = g0 - s*y - t + u;
rb=y'*x-b;
rc=C-w-x;
rtx = t.*x;
ruw = u.*w;

sigma = 0.5;  % parametro de centrado
mu = (t'*x + u'*w)/(2*n); % medida de complementaria

iter=0;

nrb = norm(rb,inf)/(1+norm(b,inf));
nre = norm(re,inf)/(1+norm(e,inf));
nrc = norm(rc,inf)/(1+norm(C,inf));
```

```
absmu = 1;

fprintf('\n            Objetivo            Objetivo
   Cifras          Longitud        Medida de           Error
   Error     Error');
fprintf('\n             primal               Dual
   Significativas      de paso     complementariedad     relativo
     relativo  relativo');
fprintf('\n iter   0.5xt*H*x - et*x   -0.5xt*H*x - Ct*u    max(-
   log10(*),0)    alpha_p            mu               ||re||     ||
   rb||    ||rc||\n');

lambda=max(10^(-5),norm(g0,'inf')); % lambda_0

while (nrb > tol || nre > tol || nrc > tol || absmu > tol)

    % Resolver el sistema con Newton.

    D = lambda + t./x + u./w;

    %Solve system
    rtx = rtx - sigma*mu;
    ruw = ruw - sigma*mu;

    V1 = -re-rtx./x+(ruw+u.*rc)./w;
    V2 = -rb;

    K1 = y.*V1./D;
    K2 = y.*y./D;

    ds = -(sum(K1)-V2)/sum(K2);
    dx = (y*ds + V1)./D;
    dw = -dx + rc;
    dt = (-rtx-t.*dx)./x;
    du = (-ruw-u.*dw)./w;
```

```
d = 1;
% Calcular alpha
alphax=(1/max(1,max(-dx./(d*x))));
alphaw=(1/max(1,max(-dw./(d*w))));
alphat=(1/max(1,max(-dt./(d*t))));
alphau=(1/max(1,max(-du./(d*u))));

alpha_p = eta*min(alphax,alphaw);
alpha_d = eta*min(alphat,alphau);

% Actualizar x,s,t,u,w

xn = x+alpha_p*dx;
s  = s+alpha_d*ds;
t  = t+alpha_d*dt;
u  = u+alpha_d*du;
w  = w+alpha_p*dw;

iter=iter+1;

% Actualizar los residuales
g = H*xn - e;
re = g - s*y - t + u;
rb=y'*xn-b;
rc=C-w-xn;
rtx = t.*xn;
ruw = u.*w;

mu = (t'*xn + u'*w)/(2*n);

lambda=lspect(xn,x,g,g0);

SV = xn./x;

g0=g;
```

```
    x=xn;

    nrb = norm(rb,inf)/(1+norm(b,inf));
    nre = norm(re,inf)/(1+norm(e,inf));
    nrc = norm(rc,inf)/(1+norm(C,inf));
    absmu = abs(mu);

    x_h_x = x'*(g0 + e);

    primal_obj = 0.5 * x_h_x - e'*x;
    dual_obj = -0.5 * x_h_x - C'*u;

    sigfig = log10(abs(primal_obj) + 1) - log10(abs(primal_obj -
        dual_obj));
    sigfig = max(sigfig, 0);

    if (iter > counter_max), status = 'ITERATION_LIMIT', break; end
    if (sigfig  > sigfig_max),  status = 'OPTIMAL_SOLUTION', break;
        end

%fprintf('%4i | %13.2e | % 16.2e | %12.3f | %17.4f | %12.2e | %17.2
    e | %.2e | %.2e\n', iter, primal_obj, dual_obj, sigfig, alpha_p,
    mu, nre, nrb, nrc);

end
fprintf('%4i | %13.2e | % 16.2e | %12.3f | %17.4f | %12.2e | %17.2e
    | %.2e | %.2e\n', iter, primal_obj, dual_obj, sigfig, alpha_p,
   mu, nre, nrb, nrc);
fobj = 0.5*x'*H*x - sum(x);
%Salida
x_stop = x;

if proceso == 1
    fprintf('\n Optimizacion terminada con exito\n');
end
```

Bajo_costo_con_identificacion.m

```
%=========================================================================
function [x_stop,iter,fobj,x_select]=bajo_costo_con_identificacion(
    ns,x,s,t,u,w,y,H,b,e,C,X)
%=========================================================================
% La funcion implementa el metodo de punto interior
% primal-dual modificado de puntos interiores para programacion
    cuadratica.
%=========================================================================
% ENTRADAS:
% ns: dimension del problema.
% x: punto inicial primal.
% s: punto inicial dual.
% t: punto inicial dual.
% u: punto inicial dual.
% w: punto inicial primal.
% y: vector de restricciones.
% H: matriz asociada a la funcion objetivo.
% b: vector asociado a las restricciones.
% e: vector asociado a la funcion objetivo.
% C: vector de cotas superiores.
%=========================================================================
% SALIDAS:
% x_stop: punto optimo.
% iter: numero de iteraciones.
% fobj: Valor en la funcion objetivo
%=========================================================================

format long
counter_max = 100000;
sigfig_max = 6.5;
proceso = 1;
tol = 1.0e-05;  % condicion de parada
eta = 0.95;     % factor para fatibilidad extricta
beta = 4;
```

```
sigfig = 0;

mmas = size(find(y==1),1);
mmenos = size(find(y==-1),1);

x_select = [];

% Calculo de los residuales
[n,m]=size(y);
g0 = H*x-e;
re = g0 - s*y - t + u;
rb=y'*x-b;
rc=C-w-x;
rtx = t.*x;
ruw = u.*w;

sigma = 0.5;  % parametro de centrado
mu = (t'*x + u'*w)/(2*n); % medida de complementaria

iter=0;

nrb = norm(rb,inf)/(1+norm(b,inf));
nre = norm(re,inf)/(1+norm(e,inf));
nrc = norm(rc,inf)/(1+norm(C,inf));

absmu = 1;

fprintf('\n            Objetivo            Objetivo
   Cifras          Longitud        Medida de           Error
   Error     Error');
fprintf('\n             primal               Dual
   Significativas      de paso     complementariedad     relativo
     relativo  relativo');
fprintf('\n iter   0.5xt*H*x - et*x   -0.5xt*H*x - Ct*u    max(-
   log10(*),0)    alpha_p            mu              ||re||     ||
   rb||    ||rc||\n');
```

```
lambda=max(10^(-5),norm(g0,'inf')); % lambda_0

while (nrb > tol || nre > tol || nrc > tol || absmu > tol)

    % Resolver el sistema con Newton.

    V1 = -re-rtx./x+(ruw+u.*rc)./w;
    V2 = -rb;

    D = lambda + t./x + u./w;

    K1 = y.*V1./D;
    K2 = y.*y./D;

    % paso predictor
    ds = -(sum(K1)-V2)/sum(K2);
    dx = (y*ds + V1)./D;
    dw = -dx + rc;
    dt = (-rtx-t.*dx)./x;
    du = (-ruw-u.*dw)./w;

    % Calcular alpfa_aff
    alphax=(1/max(1,max(-dx./x)));
    alphaw=(1/max(1,max(-dw./w)));
    alphat=(1/max(1,max(-dt./t)));
    alphau=(1/max(1,max(-du./u)));

    alpha_a = min(min(alphax,alphaw),min(alphat,alphau));

    %Calcular al gap dual afin
    mu_a = ((x+alpha_a*dx)'*(t+alpha_a*dt)+(w+alpha_a*dw)'*(u+
        alpha_a*du))/(2*n);

    %Calcular el parametros de centrado
    sigma = (mu_a/mu)^3;
```

```
if sigma>1
    sigma=0.9;
end

tau = sigma*mu;

%Resolver el sistema
rtx = rtx - tau*e + alpha_a*dx.*dt;
ruw = ruw - tau*e + alpha_a*dw.*du;

V1 = -re-rtx./x+(ruw+u.*rc)./w;
V2 = -rb;

K1 = y.*V1./D;
K2 = y.*y./D;

ds = -(sum(K1)-V2)/sum(K2);
dx = (y*ds + V1)./D;
dw = -dx + rc;
dt = (-rtx-t.*dx)./x;
du = (-ruw-u.*dw)./w;

% Calcular alpha
alphax=(1/max(1,max(-dx./x)));
alphaw=(1/max(1,max(-dw./w)));
alphat=(1/max(1,max(-dt./t)));
alphau=(1/max(1,max(-du./u)));

alpha_p = eta*min(alphax,alphaw);
alpha_d = eta*min(alphat,alphau);

% Actualizar x,s,t,u,w

xn = x+alpha_p*dx;
s  = s+alpha_d*ds;
```

```
t  = t+alpha_d*dt;
u  = u+alpha_d*du;
w  = w+alpha_p*dw;

iter=iter+1;

% Actualizar los residuales
g = H*xn - e;
re = g - s*y - t + u;
rb=y'*xn-b;
rc=C-w-xn;
rtx = t.*xn;
ruw = u.*w;

mu = (t'*xn + u'*w)/(2*n);

lambda=lspect(xn,x,g,g0);

SV = xn./x;

g0=g;
x=xn;

ww = (w.*x)./(x.*u +w.*t);

if sigfig > 1

    rho = mu^(1/beta);
    theta = 100;
    Cqu = size(find(SV>0.9),1);

    % Y = + 1  , AZULES
    Cql1 = find(ww(y==1) >= theta*sqrt(mu));

    qmas = max(size(Cql1,1),min(ceil(min(ceil(rho*ns),Cqu)/2),
        mmas));
```

```
        % q+ = max(ql+,min(|min(rho*m,qu)/2|,m+)), m : numero de
            datos ns, m+:
        % numero de datos positivos

        qq_aux_pos=[];
        for ii=1:qmas
            mas_grande = find(ww==max(ww(y==1)));
            qq_aux_pos(ii)=mas_grande(1);
            ww(mas_grande(1))=0;
        end

        % Y = - 1 , ROJOS
        Cql2 = find(ww(y==-1) >= theta*sqrt(mu));

        qmenos = max(size(Cql2,1),min(ceil(min(ceil(rho*ns),Cqu)/2)
            ,mmenos));

        qq_aux_neg=[];
        for ii=1:qmenos
            mas_grande = find(ww==max(ww(y==-1)));
            qq_aux_neg(ii)=mas_grande(1);
            ww(mas_grande(1))=0;
        end

        if ~isempty(size(qq_aux_pos)) && ~isempty(size(qq_aux_neg))
            x_select = [qq_aux_pos,qq_aux_neg];
            break
        end

        end


    nrb = norm(rb,inf)/(1+norm(b,inf));
    nre = norm(re,inf)/(1+norm(e,inf));
    nrc = norm(rc,inf)/(1+norm(C,inf));
```

```
    absmu = abs(mu);

    x_h_x = x'*(g0 + e);

    primal_obj = 0.5 * x_h_x - e'*x;
    dual_obj = -0.5 * x_h_x - C'*u;

    sigfig = log10(abs(primal_obj) + 1) - log10(abs(primal_obj -
        dual_obj));
    sigfig = max(sigfig, 0);

    if (iter > counter_max), status = 'ITERATION_LIMIT', break; end
    if (sigfig  > sigfig_max),  status = 'OPTIMAL_SOLUTION', break;
        end

    %fprintf('%4i | %13.2e | % 16.2e | %12.3f | %17.4f | %12.2e |
        %17.2e | %.2e | %.2e\n', iter, primal_obj, dual_obj, sigfig,
        alpha_p, mu, nre, nrb, nrc);

end
fobj = 0.5*x'*H*x - sum(x);
%Output
x_stop = x;

fprintf('%4i | %13.2e | % 16.2e | %12.3f | %17.4f | %12.2e | %17.2e
    | %.2e | %.2e\n', iter, primal_obj, dual_obj, sigfig, alpha_p,
   mu, nre, nrb, nrc);

if isempty(x_select) x_select = 0; end

if proceso == 1
    fprintf('\n Identificacion terminada\n');
end
```

APÉNDICE B

CODIGO DEL PROGRAMA EN C/C++

En el archivo *svm.cpp* anexamos nuestros algoritmo de bajo costo sin paso predictor tal como se explica a continuación.

svmtrain_mpi.cpp

```
void Solver::SolveIPM(int l, const QMatrix& Q,
            const double *p_, const schar *y_,
            double *alpha_, double Cp,
            double Cn, double eps,
                    SolutionInfo* si, int shrinking)
{
this->l = l;
this->Q = &Q;
QD=Q.get_QD();
clone(p, p_,l);
clone(y, y_,l);
clone(alpha,alpha_,l);
this->Cp = Cp;
this->Cn = Cn;
this->eps = eps;
unshrink = false;

double eta = 0.95;              // Numero que multiplica a la
   longitud de paso alfa
double tau = 0.0;               // Camino central
double sigma = 0.5;             // Parametro de centrado
double mu = 0.0;                // Medida de complementariedad
```

```
double sum_rtx = 0.0;           // Suma de los residuales t*x
double sum_ruw = 0.0;           // Suma de los residuales u*w
double rb = 0.0;                // Residual rb
double re = 0.0;                // Residual re
double rc = 0.0;                // Residual rc
double sum_K1 = 0.0;            // Suma para la direccion ds
double sum_K2 = 0.0;            // Suma para la direccion ds
double V2;                      // Valor para resolver el sistema
    de ecuaciones
double alfa_p = 0.0;            // Longitud de paso de la variale
    primal
double alfa_d = 0.0;            // Longitud de paso de la variable
    dual
double nrb = 0.0;               // norma nrb para la condicion de
    parada
double nre = 0.0;               // norma nre para la condicion de
    parada
double nrc = 0.0;               // norma nrc para la condicion de
    parada
double lambda = 0.0;            // Longitud espectral
double lmin = 1e-8;             // Longitud espectral lambda_min
double sts = 0.0;
double sty = 0.0;
double ss = 0.0;                // Para la longitud espectral

// condiciones de parada
double sigfig_max = 6.5;        // parar por sifras significativas
double tol = 1e-5;              // Condicion de parada
int counter_max = 100000;       // parar si supera el limite de
    iteraciones
double sigfig = 0.0;
double primal_obj = 0.0;
double dual_obj = 0.0;
double x_h_x = 0.0;

double norm_re = 0.0;
```

```
double norm_rc = 0.0;

// Variables primales
// alpha, w
double *w;

// Variable duales
double s = 0.0;
double *t;
double *u;

// Direccion del metodo de newton
double ds = 0.0;
double *dx;                    // se refiere a la direccion de
    alpha
double *dt;
double *du;
double *dw;
double *identi;

double *ruw;
double *rtx;
double *diag;                  // Diagonal que contiene la
    longitud espectral

// Otros
G = new double[l];             // gradiente(alpha)
G_bar = new double[l];         // gradiente(alpha_new)
ruw = new double[l];           // Residual ruw
rtx = new double[l];           // Residual rtx
diag = new double[l];          // Diagonal del sistema de
    ecuaciones
dx = new double[l];
dt = new double[l];
du = new double[l];
dw = new double[l];
```

```
w = new double[l];
t = new double[l];
u = new double[l];
identi = new double[l];

// tiempo
double co, c1;
co =  clock();

int i,j;



// initialize alpha_status
{
alpha_status = new char[l];
for(int i=0;i<l;i++)
update_alpha_status(i);
}

// initialize active set (for shrinking)
{
active_set = new int[l];
for(int i=0;i<l;i++)
active_set[i] = i;
active_size = l;
}



//Calcular residuales re,rc,rb,rtx,tuw,G gradiente y punto inicial
{
for(i=0;i<l;i++)
{
rtx[i] = 0.0;// inicializar en 0.0
ruw[i] = 0.0;

alpha[i]=get_C(i);
```

```
w[i]=get_C(i);
t[i]=get_C(i);
u[i]=get_C(i);

if(get_C(i)>1){
t[i]=1/get_C(i);
u[i]=1/get_C(i);
}

G[i] = -1;
G_bar[i] = 0;
}
for(i=0;i<l;i++)
{
const Qfloat *Q_i = Q.get_Q(i,l);
double alpha_i = alpha[i];
for(j=0;j<l;j++){
G[j] += alpha_i*Q_i[j];
}
}
}

norm_re = 0.0;
norm_rc = 0.0;
for(i=0;i<l;i++)
{
re = G[i] -s*y[i] - t[i] + u[i];
rc = get_C(i) - w[i] - alpha[i];
ruw[i] = u[i] * w[i];
rtx[i] = t[i] * alpha[i];
rb += y[i] * alpha[i];
sum_rtx += rtx[i];
sum_ruw += ruw[i];
norm_re = max(norm_re,fabs(re));
norm_rc = max(norm_rc,fabs(rc));
}
```

```
mu = (sum_rtx + sum_ruw)/(2*l);

// normas de vectores

nrb = fabs(rb);
nrc = norm_rc/(1 + Cp);
nre = norm_re/(2);

int iter = 0;

lambda = max(1e-5,norma_inf(G));

while(1)
{
// Resolver el sistema con newton

V2 = -rb;
sum_K2 = 0.0;
for(i=0;i<l;i++)
{
diag[i] = lambda + t[i]/alpha[i] + u[i]/w[i];
sum_K2 += 1/diag[i];
}

tau = sigma*mu;

// Resolver el sistema con newton

sum_K1 = 0.0;
for(i=0;i<l;i++)
{// Actualizamos con el paso central

rtx[i] = rtx[i] - tau;
ruw[i] = ruw[i] - tau;
re = G[i] -s*y[i] - t[i] + u[i];
```

```
rc = get_C(i) - w[i] - alpha[i];
sum_K1 += y[i] * (-re - rtx[i]/alpha[i] + (ruw[i] + u[i]*rc)/w[i])/
    diag[i];
}

ds = -(sum_K1 - V2)/sum_K2;

// Busqueda lineal para las variables primal dual
alfa_p = 1.0;
alfa_d = 1.0;
for(i=0;i<l;i++)
{
re = G[i] -s*y[i] - t[i] + u[i];
rc = get_C(i) - w[i] - alpha[i];
dx[i] = (y[i] * ds + (-re - rtx[i]/alpha[i] + (ruw[i] + u[i]*rc)/w[
    i]))/diag[i];
if (dx[i] < 0.0) alfa_p = min(alfa_p,-alpha[i]/dx[i]);
dw[i] = - dx[i] + rc;
if (dw[i] < 0.0) alfa_p = min(alfa_p,-w[i]/dw[i]);
dt[i] = (-rtx[i] - t[i] * dx[i])/alpha[i];
if (dt[i] < 0.0) alfa_d = min(alfa_d,-t[i]/dt[i]);
du[i] = (-ruw[i] - u[i] * dw[i])/w[i];
if (du[i] < 0.0) alfa_d = min(alfa_d,-u[i]/du[i]);
}

alfa_p = eta*min(alfa_p,1.0);
alfa_d = eta*min(alfa_d,1.0);

// Actualizar x,s,t,u,w
s = s + alfa_d*ds;

// Actualizar residuales


for(i=0;i<l;i++)
{
```

```
alpha_[i] = alpha[i] + alfa_p*dx[i];
w[i] += alfa_p*dw[i];
t[i] += alfa_d*dt[i];
u[i] += alfa_d*du[i];
G_bar[i] = -1;
}

{
for(i=0;i<l;i++)
if(alpha_[i] > 0.0000001)
{
const Qfloat *Q_i = Q.get_Q(i,l);
double alpha_i = alpha_[i];
for(j=0;j<l;j++)
G_bar[j] += alpha_i*Q_i[j];
}
}

rb = 0.0;
// parametros de centrado y complementariedad
sum_rtx = 0.0;
sum_ruw = 0.0;
// longitud espectral
sts = 0.0;
sty = 0.0;
norm_re = 0.0;
norm_rc = 0.0;

for(i=0;i<l;i++)
{
re = G_bar[i] -s*y[i] - t[i] + u[i];
rc = get_C(i) - w[i] - alpha_[i];
ruw[i] = u[i] * w[i];
rtx[i] = t[i] * alpha_[i];
rb += y[i] * alpha_[i];
sum_rtx += rtx[i];
```

```
sum_ruw += ruw[i];
ss = alpha_[i] - alpha[i];
sts += ss*ss;
sty += ss*(G_bar[i] - G[i]);
identi[i] = alpha_[i]/alpha[i];
// Regresamos la solucion
alpha[i] = alpha_[i];
G[i] = G_bar[i];
norm_re = max(norm_re,fabs(re));
norm_rc = max(norm_rc,fabs(rc));
}

mu = (sum_rtx + sum_ruw)/(2*l);

lambda = max(lmin,sty/sts);

if(isnan(lambda)) lambda = lmin;

// normas de vectores

nrb = fabs(rb);
nrc = norm_rc/(1 + Cp);
nre = norm_re/(2);

++iter;

/* instrumentation */
x_h_x = 0;
primal_obj = 0;
dual_obj = 0;

for (i=0; i<l; i++) {
x_h_x += (G[i]+1)*alpha[i];
}

primal_obj = 0.5 * x_h_x;
```

```
dual_obj = -0.5 * x_h_x;

for (i=0; i<l; i++) {
primal_obj += alpha[i]*(-1);
dual_obj += u[i]*(-Cp);
}

sigfig = log10(ABS(primal_obj) + 1) - log10(ABS(primal_obj -
    dual_obj));
sigfig = max(sigfig, 0.0);

if (iter > counter_max) { info("\n Limite de iteraciones superado")
    ; break;}
if (sigfig  > sigfig_max) { break; }
if (nrb <= tol && nrc <= tol && nre <= tol && abs(mu) <= tol) break
    ;

}
// fin ciclo

c1 = clock();

info("tiempo %f\n",(float) (c1 -co)/CLOCKS_PER_SEC);

for(i=0;i<l;i++){
if(alpha[i] >= get_C(i)-0.01){
alpha_status[i] = UPPER_BOUND;
alpha[i]=get_C(i);
}
else if(identi[i] < 0.89 || alpha[i]<tol){
alpha_status[i] = LOWER_BOUND;
alpha[i]=0;
}
else alpha_status[i] = FREE;

//info("alfa[%d] = %f , identi[%d] = %f\n",i,alpha[i],i,identi[i]);
```

```
//getchar();
}

info("\niter = %d, |y'x| = %f ||rc|| = %f ||re|| = %f\n",iter,nrb,
    nrc,nre);

//put back the solution
{
for(int i=0;i<l;i++)
alpha_[i] = alpha[i];
}

// calculate rho

si->rho = calculate_rho();

// calculate objective value
{
double v = 0;
int i;
for(i=0;i<l;i++)
v += alpha[i] * (G[i] + p[i]);

si->obj = v/2;
}

si->upper_bound_p = Cp;
si->upper_bound_n = Cn;

info("\noptimization finished, #iter = %d\n",iter);

delete[] G;
delete[] G_bar;
delete[] ruw;
delete[] rtx;
delete[] diag;
```

```
delete[] dx;
delete[] dt;
delete[] du;
delete[] dw;
delete[] p;
delete[] y;
delete[] alpha;
delete[] alpha_status;
delete[] active_set;
delete[] w;
delete[] t;
delete[] u;
delete[] identi;

}
```

BIBLIOGRAFÍA

[1] P. Arbenz and Z. Drmac. On positive semidefinite matrices with known null space. *SIAM J. Matrix Anal. Appl*, 24:132–149, 2000.

[2] A. S. E. Bakry, R. A. Tapia, and Y. Zhang. A study of indicators for identifying zero variables in interior-point methods. *SIAM Review*, 36(1):45–72, 1994.

[3] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[4] M. S. Bazaraa and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, New York, 1979.

[5] E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10:1196–1211, 2000.

[6] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[8] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, UK, 2000.

[9] Y.-H. Dai, W. W. Hager, K. Schittkowski, and H. Zhang. The cyclic Barzilai–Borwein method for unconstrained optimization. *IMA J Numer Anal*, 26(3): 604–627, 2006.

[10] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[11] S. Fine, K. Scheinberg, N. Cristianini, J. Shawe-taylor, and B. Williamson. Efficient svm training using low-rank kernel representations. *Journal of Machine*

Learning Research, 2:243–264, 2001.

[12] W. Glunt, T. L. Hayden, and M. Raydan. Molecular conformations from distance matrices. *Journal Computational Chemistry*, 14:114–120, 1993.

[13] M. D. González-Lima, W. W. Hager, and H. Zhang. An affine-scaling interior-point method for continuous knapsack constraints with application to support vector machines. *SIAM Journal on Optimization*, 21(1):361–390, 2011.

[14] N. Gould and S. Leyffer. An introduction to algorithms for nonlinear optimization. In *Frontiers in Numerical Analysis*. Springer Berlin Heidelberg, 2003.

[15] G. Jabbour, R. Márquez, L. Ruiz, and L. Maldonado. Reconocimiento de firmas off-line mediante máquinas de vectores de soporte. *Ciencia e Ingeniería*, 31(1): 43–52, 2011.

[16] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142. Springer Verlag, Heidelberg, DE, Chemnitz, DE, 1998.

[17] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *in Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

[18] T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[19] J. H. Jung, D. P. Oleary, Andr, and L. Tits. Adaptive constraint reduction for convex quadratic programming. In *In Preparation*, 2007.

[20] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Comput.*, 13(3):637–649, 2001.

[21] T. Knebel, S. Hochreiter, and K. Obermayer. An smo algorithm for the potential support vector machine. *Neural Computation*, 20(1):271–287, 2008.

[22] C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:1288–1298, 2001.

[23] W. Liu and Y. H. Dai. Minimization algorithms based on supervisor and searcher cooperation. *Journal Optimization Theory and Applications*, 111(2): 359–379, 2001.

[24] S. Maldonado. *Utilización de Support Vector Machines no lineal y selección de atributos para credit scoring.* Tesis de maestria en gestión de operaciones, Universidad de Chile, Santiago de Chile, 2007.

[25] MATLAB. *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.

[26] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[27] J. Nocedal and S. Wright. *Numerical optimization.* Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

[28] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[29] J. Platt. Fast training of svms using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning.* MIT Press, Cambridge, Mass, 1998.

[30] B. Schölkopf and A. J. Smola. Learning with kernel: support vector machines, regularization, optimization, and beyond. MIT Press, 2002.

[31] T. Serafini and L. Zanni. On the working set selection in gradient-based decomposition techniques for support vector machines. *Optimization Methods and Software*, 2005.

[32] T. Serafini, G. Zanghirati, and L. Zanni. Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20(2):353–378, 2005.

[33] V. Vapnik. *Statistical learning theory.* Wiley, 1998.

[34] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[35] S. J. Wright. *Primal-dual interior-point methods.* SIAM, 1997.

[36] L. Zanni. An improved gradient projection-based decomposition techniques for support vector machines. *Computational Management Science*, 3:131–145, 2006.