Revista Politécnica ISSN 1900-2351 (Impreso), ISSN 2256-5353 (En línea), Año 21, Número 42, páginas 99-121, Julio-Diciembre 2025

DOI: https://doi.org/10.33571/rpolitec.v21n42a7

LA MÁQUINA DE SOPORTE VECTORIAL COMO PROBLEMA DE PROGRAMACIÓN CUADRÁTICA: ANÁLISIS Y UN TUTORIAL

Carlos Alberto Henao-Baena

Magister en Ingeniería Eléctrica, Ejecutor Técnico Programa Talento Tech, Universidad de Caldas. caralbhenao@utp.edu.co, https://orcid.org/0000-0001-9873-8211

RESUMEN

La transformación del problema de optimización en su forma dual a su equivalencia matricial en una máquina de soporte vectorial (SVM) es una etapa crucial en el desarrollo matemático del algoritmo. Este proceso es clave para lograr una correcta implementación de la SVM. Sin embargo, los detalles matemáticos de esta transformación suelen no estar documentados. Una exploración a mayor grado de profundidad permite identificar las operaciones matriciales y vectoriales involucradas, estableciendo la relación de la SVM con un problema de programación cuadrático. En este artículo, se identifican las operaciones vectoriales necesarias para transformar el problema de optimización en su forma dual a su notación matricial, determinado así la analogía entre la SVM y un QP. Se examinan los algoritmos C-SVM y R-SVM desarrollando el componente matemático completo. Finalmente, se implementa un caso de estudio para validar el desarrollo propuesto..

Palabras clave: Máquina de Soporte Vectorial, Programación Cuadrática, Clasificación, Regresión, Aprendizaje de Máquinas.

Recibido: 29 de septiembre de 2025. Aceptado: 11 de noviembre de 2025
Received: September 29, 2025. Accepted: November 11, 2025

THE SUPORT VECTOR MACHINE AS A QUADRATIC PROGRAMMING PROBLEM: ANALYSIS AND TUTORIAL

ABSTRACT

The transformation of the optimization problem in its dual form to its matrix equivalent in a Support Vector Machine (SVM) is a fundamental stage in the mathematical development of the algorithm. This process is crucial to achieve a correct implementation of the SVM. However, the mathematical details of this transformation are often not documented. A deeper exploration allows to identify the matrix and vector operations, establishing the relationships between the SVM and a Quadratic Programming problem (QP). In this article, the basic vectorial operations are identified to transform the optimization problem in its dual form to its matrix notation, determining the analogy between the SVM and a QP. The C-SVM and R-SVM algorithms are examined, fully developing the mathematical component. Finally, a case study is conducted to validate the proposal development.

Keywords: *Suport Vector Machine, Quadratic Programming, Classification, Regresion, Machine Learning*

Cómo citar este artículo: C. Henao. "La máquina de soporte vectorial como problema de programación cuadrática: análisis y un tutorial", Revista Politécnica, vol.21, no.42 pp.99-121, 2025. DOI:10.33571/rpolitec.v21n42a7

1. INTRODUCCIÓN

Las máquinas de soporte vectorial (SVM) [1] son uno de los algoritmos más ampliamente utilizado en aprendizaje de máquinas para tareas de regresión y clasificación, desde detección y diagnóstico de enfermedades hasta aplicaciones en agricultura de precisión [2,3]. Un aspecto clave de las SVM es su relación con un problema de programación cuadrática (QP), generalmente con restricciones lineales, donde la convexidad garantiza que la solución obtenida corresponda a un mínimo global [4].

El modelo matemático de una SVM generalmente consta de tres elementos, la función de costo, las restricciones de igualdad y las desigualdades. Por ejemplo, los algoritmos C-SVM (SVM en clasificación) y R-SVM (SVM en regresión) contienen los tres elementos en su formulación [1,5]. Otros algoritmos no consideran las restricciones de desigualdad en el modelo matemático de la SVM, esto conlleva a que la solución no requiera necesariamente de métodos numéricos iterativos para calcular la solución cuando las igualdades son lineales, por ejemplo, los algoritmos LS-SVM [6], PR-SVM [7], LSTW-SVM [8], KBLSTW-SVM y WLSTW-SVM utilizan la solución de mínimos cuadrados o la solución regularizada o ridge para determinar los parámetros (vectores de soporte) del modelo [9,10,11,12]. Sin embargo, se reportan algoritmos que utiliza métodos iterativos para refinar o computar la solución [13]. Por otro lado, los algoritmos que contienen desigualdades suelen requerir métodos iterativos para encontrar la solución, debido principalmente a la formulación de las SVM como un problema de programación cuadrática. Una gran cantidad de algoritmos pueden encontrase bajo este paradigma en la literatura, algunos de estos, B-SVM [14], V-SVM[15], TW-SVM[16], CS-SVM[17].

Gran parte de los algoritmos de SVM están disponibles en herramientas computacionales o en bibliotecas de código abierto para distintos lenguajes de programación. LS-SVMlab, escrita en lenguaje C, es una herramienta para Matlab inspirada en el LS-SVM [18-19]. Otras herramientas computacionales para Matlab implementan diferentes variantes del algoritmo SVM, como C-SVM [20], V-SVM, R-SVM [21,22]. Otras bibliotecas están desarrolladas para Python, siendo Scikit-learn una de las más populares. En esta se pueden implementar diferentes versiones de la SVM, como V-SVM, C-SVM entre otros [23]. Generalmente, al construir una SVM utilizando las bibliotecas disponibles, se realiza definiendo el modelo, los datos o características de entrenamiento, los parámetros e hiper-parámetros y los datos de validación. Los procesos adicionales, como la formulación del problema de programación cuadrática, la construcción de kernel, el cálculo de la predicción, entre otros procesos, suelen estar ocultos al usuario, que no requiere un conocimiento avanzado sobre las SVM. Sin embargo, en algunas situaciones puede ser necesario modificar o implementar un algoritmo no disponible en las bibliotecas, lo cual requerirá de un conocimiento profundo sobre el procedimiento de implementación de una SVM.

El proceso de formulación y solución de una SVM presenta varias etapas, en síntesis, en el primer paso se diseña el algoritmo de regresión o clasificación como un problema optimización, definiendo la función de costo y las restricciones correspondientes. Luego, este problema de optimización primario se transforma en su forma dual, donde las incógnitas son los multiplicadores de Lagrange, generalmente la representación dual se presenta en notación sigma. En tercer lugar, el problema dual se formula como un QP, para ello, el problema dual debe desarrollarse en una notación matricial para identificar los parámetros del QP, aplicando operaciones sobre vectores y matrices para sustituir la notación sigma por una vectorial. Finalmente, los parámetros de la SVM se determinan resolviendo el QP.

La transformación del problema de optimización en su forma dual a su equivalencia matricial es una etapa crucial en el desarrollo matemático del algoritmo. Generalmente, esta transformación se presenta sin detallar en profundidad las operaciones vectoriales necesarias para llevarla a cabo [1,4,5,14], por ejemplo, en [24] se desarrolla la formulación matemática matricial al problema de clasificación en una SVM, no obstante, los autores no detallan los pasos matemáticos intermedios que permiten comprender las transiciones desde la formulación escalar (basada en sumatorias) hasta su equivalente matricial cuadrático. Conocer este procedimiento en detalle permitirá identificar las operaciones sobre vectores y matrices que se deben considerar para establecer la relación entre una SVM con el QP. En este artículo se estudia a profundidad la relación entre la SVM y el QP. Se parte del problema de optimización en su forma dual, y luego se deducen las expresiones vectoriales que permiten establecer la relación entre ambos problemas. Estas operaciones hacen posible rescribir todo el modelo matemático de la SVM en notación vectorial, lo que permite observar la analogía entre la SVM y el QP. Se analizan dos algoritmos, C-SVM y R-SVM, con los cuales se desarrolla todo el marco matemático y se establece un marco comparativo con el QP. Por último, se plantea un caso de estudio que valida el desarrollo matemático propuesto compatible con Matlab y Python. Los aportes de este artículo se resumen a continuación.

1. Se identifican las operaciones vectoriales necesarias para transformar el problema de optimización en su forma dual a su notación matricial estándar, a partir del análisis de los algoritmos C-SVM y R-SVM en su notación sigma [4].
2. Se analiza la relación entre un QP y los algoritmos C-SVM y R-SVM, destacando cómo las expresiones vectoriales permiten establecer tal relación.

El documento está organizado de la siguiente manera. La primera parte introduce el marco metodológico, comenzando con el desarrollo matemático y la identificación de las operaciones vectoriales necesarias para transformar los algoritmos C-SVM y R-SVM en una forma vectorial. A continuación, se introduce el QP y se establece su relación con los algoritmos C-SVM y R-SVM. El marco metodológico concluye definiendo un caso de estudio para validar el desarrollo matemático propuesto en este estudio. En la parte final, se comparten los resultados del proceso de validación, por último, las conclusiones finales del estudio.

2. MATERIALES Y METODOGIA

Tabla 1: Nomenclatura adoptada

Término	Descripción
$\boldsymbol{t}$	Vector de etiquetas de salida de entrenamiento
$\boldsymbol{y}$	Vector de etiquetas de salida de validación
$\boldsymbol{x}$	Matriz de características o entrada de entrenamiento
$\boldsymbol{z}$	Matriz de características o entrada de validación
$y(\boldsymbol{z})$	Predicción sobre la entrada $\boldsymbol{z}$
$\boldsymbol{a}$	Vector de multiplicadores de Langrage caso clasificación
$\ddot{\boldsymbol{a}}$	Vector de multiplicadores de Langrage caso regresión
$\boldsymbol{K_{xx}}$	Matriz Gram o de kernel para la entrada $\boldsymbol{x}$
$\boldsymbol{K_{xz}}$	Matriz Gram o de kernel entre las entradas $\boldsymbol{x}$ y $\boldsymbol{z}$
b	Bias o valor medio de la predicción
$C, \varphi, \frac{1}{2\sigma^2}$	Parámetros e hiper-parámetros del modelo

A continuación, se presenta el marco metodológico implementado. En las primeras secciones se describe en detalle el desarrollo matemático para transformar la ecuación Lagrangiana en su forma dual, expresada en notación sigma, en su equivalente vectorial. Se toman como punto de partida las ecuaciones Lagrangianas, tanto para un caso de clasificación binaria y regresión con una única salida sobre los algoritmos C-SVM y R-SVM. Luego, se analiza la estrecha relación entre un paradigma de programación cuadrática y la estructura vectorial de la SVM, identificando las equivalencias entre ambas representaciones. Finalmente, se detallan los instrumentos metodológicos para implementar la SVM en un caso de estudio, defiendo los datos, los parámetros e hiper-parámetros de prueba, las métricas de validación, el pseudocódigo de prueba, así como las herramientas computacionales utilizadas en este trabajo. La Tabla 1 muestra la nomenclatura adoptada, así como el significado de cada elemento de la notación.

2.1. La SVM para clasificación C-SVM: un modelo básico como un QP

La forma dual de la función Lagrangiana para una SVM en clasificación viene dada por la siguiente expresión (ecuación 1) [4].

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m)$$

$$0 \leq a_n \leq C \tag{1}$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

Donde $\boldsymbol{x} = (x_1 \quad \cdots \quad x_N)^\top$ corresponde a la entrada y $\boldsymbol{t} = (t_1 \quad \cdots \quad t_N)^\top$ las respectivas etiquetas en las salidas con $t_n \in \{-1, 1\}$, tanto $\boldsymbol{x}$ cómo $\boldsymbol{t}$ se utilizan para entrenar el modelo. Es de aclarar que cada uno de los valores de $\boldsymbol{x}$ puede ser vectores de j componentes $x_i = (x_{i1} \quad \cdots \quad x_{ij})^\top$ con $i \leq N$), por lo que $\boldsymbol{x}$ será una matriz $\boldsymbol{x} \in \mathbb{R}^{N \times j}$, por otro lado, definimos el kernel como $k(x, x^\top) = \emptyset(x)\emptyset(x^\top)$ donde $\emptyset(x)$ es la función base, la cual realiza una transformación de la entrada $\boldsymbol{x}$ a un nuevo espacio de características. Definimos el vector $\boldsymbol{a} = (a_1 \quad \cdots \quad a_N)^\top$ como los multiplicadores de Lagrange de la función Lagrangiana dual $\tilde{L}$ definida en la ecuación 1. En síntesis, la fase de entrenamiento de una SVM consiste en determinar el vector $\boldsymbol{a}$. Para determinar el vector $\boldsymbol{a}$, se resuelve el problema de optimización restringido que se presenta en la ecuación 2 [1,4].

$$\begin{gathered} \max_{\boldsymbol{a}} \tilde{L}(\boldsymbol{a}) \\ sujeto: \\ 0 \leq a_n \leq C \\ \sum_{n=1}^{N} a_n t_n = 0 \end{gathered} \tag{2}$$

El problema descrito en la ecuación 2 tiene como objetivo determinar el vector $\boldsymbol{a}$ que maximiza la función Lagrangiana dual. La solución se encuentra restringida por $2N + 1$ condiciones, en base a las $2N$ desigualdades expresadas en el término $0 \leq a_n \leq C$ y la igualdad $\sum_{n=1}^{N} a_n t_n = 0$. Una vez el modelo esté entrenado, podremos evaluarlo para clasificar un nuevo vector de entrada. La estructura del modelo predictivo tendrá la siguiente forma (ver ecuación 3) [5].

$$y(z) = sign\left\{\sum_{n=1}^{N} a_n t_n k(z, x_n) + b\right\} \tag{3}$$

Donde z es el valor de entrada $z_i = (z_{i1} \quad \cdots \quad z_{ij})^\top$, $y(z)$ es la predicción de la SVM en z. Por otro lado, b es el *bias*. Para determinar el valor del parámetro b se suele utilizar el promedio entre las predicciones realizadas sobre los datos de entrenamiento $\boldsymbol{x}$ y las etiquetas $\boldsymbol{t}$ (ver ecuación 4) [4].

$$b = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - \sum_{m=1}^{N} a_m t_m k(x_n, x_m)\right) \tag{4}$$

La solución encontrada al resolver el problema presentado en la ecuación 2 podrá arrojar valores tales que $a_n = 0$, es decir, multiplicadores de Lagrange que no contribuyen a la predicción (ecuación 4). Lo anterior se explica al plantear las condiciones de Karush – Kuhn – Tucker (KKT) para $\tilde{L}(\boldsymbol{a})$ de la ecuación 1, estas se enseñan en la ecuación 5, básicamente estas establecen que si $a_n > 0$ entonces $t_n y(x_n) - 1 = 0$ y viceversa. Los valores de $a_n \neq 0$ se conocen como vectores de soporte de la SVM o soluciones sparse y corresponden a los multiplicadores de Lagrange asociados al dato x_n que aporta información crucial para la construcción de la SVM [4].

$$\begin{gathered} a_n \geq 0 \\ t_n y(x_n) - 1 \geq 0 \\ a_n(t_n y(x_n) - 1) = 0 \end{gathered} \tag{5}$$

2.2. La SVM para clasificación como un QP

Ahora se desarrolla el procedimiento que explica cómo reescribir la ecuación 1 de una manera vectorial. Lo primero que se analiza es el término que contiene la doble suma en la ecuación 1, para esto se expande las sumas como se presenta en la ecuación 6.

$$\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) = \sum_{n=1}^{N} a_n t_n \left(a_1 t_1 k(x_n, x_1) + \cdots + a_N t_N k(x_n, x_N)\right)$$

$$= \sum_{n=1}^{N} a_n \left(a_1 t_n t_1 k(x_n, x_1) + \cdots + a_N t_n t_N k(x_n, x_N)\right) \tag{6}$$

Notar que todos los productos $t_n t_1$ hasta $t_n t_N$ pueden ser expresados utilizando un producto externo o tensorial del vector $\boldsymbol{t}$ (ver ecuación 7).

$$\begin{pmatrix} t_1 \\ t_2 \\ \cdot \\ \cdot \\ t_N \end{pmatrix} (t_1 \quad t_2 \quad \cdot \quad \cdot \quad \cdot \quad t_N) = \begin{pmatrix} t_1^2 & t_1 t_2 & \cdots & t_1 t_N \\ t_2 t_1 & t_2^2 & \cdots & t_2 t_N \\ \vdots & \vdots & \ddots & \vdots \\ t_N t_1 & t_N t_2 & \cdots & t_N^2 \end{pmatrix} = \boldsymbol{t}\boldsymbol{t}^{\top} \tag{7}$$

Ahora resulta necesario analizar los productos $t_n t_1 k(x_n, x_1)$ hasta $t_n t_N k(x_n, x_N)$. Aquí se presenta N^2 productos, que coincide con el número de elementos de la matriz definida en la ecuación 7. Notemos que, si se requiere desarrollar el producto $t_n t_m k(x_n, x_m)$, se debe multiplicar cada elemento del producto externo $\boldsymbol{t}\boldsymbol{t}^{\top}$ por cada elemento de una matriz que contenga la información de $k(x_n, x_m)$. El producto elemento a elemento entre matrices se conoce como producto Hadamard, y se denota con el símbolo $\odot$ [25]. Consideremos el producto Hadamard expresado en la ecuación 8.

$$\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}} = \begin{pmatrix} t_1^2 & \cdots & t_1 t_N \\ \vdots & \ddots & \vdots \\ t_N t_1 & \cdots & t_N^2 \end{pmatrix} \odot \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{pmatrix}$$

$$\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}} = \begin{pmatrix} t_1^2 k(x_1, x_1) & \cdots & t_1 t_N k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ t_N t_1 k(x_N, x_1) & \cdots & t_N^2 k(x_N, x_N) \end{pmatrix} \tag{8}$$

Con

$$\boldsymbol{K}_{\boldsymbol{xx}} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{pmatrix}$$

Donde $\boldsymbol{K}$ es la matriz de Gram o de kernel, en este caso $\boldsymbol{K}_{\boldsymbol{xx}}$ hace referencia a la matriz kernel sobre la matriz $\boldsymbol{x}$. Notar que la matriz $\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}}$ reúne todas las operaciones $t_n t_m k(x_n, x_m)$. Ahora nos concentramos en la suma $a_1 t_n t_1 k(x_n, x_1) + \cdots + a_N t_n t_N k(x_n, x_N)$ de la ecuación 6. Analicemos el caso cuando $n = N$. En tal caso, la suma se transforma en $a_1 t_N t_1 k(x_N, x_1) + \cdots + a_N t_N^2 k(x_N, x_N)$. Notar que los términos $t_N t_m k(x_N, x_m)$ corresponden a la última fila de la matriz $\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}}$. Por tanto, para completar el término, se debe multiplicar cada elemento de la fila N de la matriz de la ecuación 8 por el respectivo multiplicador de Lagrange a_m. Notar que esto se puede realizar a través de un producto matricial entre la matriz $\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}}$ y el vector $\boldsymbol{a}$, tal como se enseñan en la ecuación 9.

$$(\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}})\, \boldsymbol{a} = \begin{pmatrix} t_1^2 k(x_1, x_1) & \cdots & t_1 t_N k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ t_N t_1 k(x_N, x_1) & \cdots & t_N^2 k(x_N, x_N) \end{pmatrix} \begin{pmatrix} a_1 \\ \cdot \\ a_N \end{pmatrix} \tag{9}$$

$$(\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}})\, \boldsymbol{a} = \begin{pmatrix} a_1 t_1^2 k(x_1, x_1) + \cdots + a_N t_1 t_N k(x_1, x_N) \\ \cdot \\ a_1 t_N t_1 k(x_N, x_1) + \cdots + a_N t_N^2 k(x_N, x_N) \end{pmatrix}$$

De esta manera, los términos $a_1 t_n t_1 k(x_n, x_1) + \cdots + a_N t_n t_N k(x_n, x_N)$ quedan incluidos en la operación $(\boldsymbol{t}\boldsymbol{t}^\top \odot \boldsymbol{K_{xx}})\boldsymbol{a}$. Ahora se realiza el siguiente cambio de variable (ver ecuación 10).

$$\alpha_n = a_1 t_n t_1 k(x_n, x_1) + \cdots + a_N t_n t_N k(x_n, x_N) \tag{10}$$

De este modo, la ecuación 6 se transforma en $\sum_{n=1}^{N} a_n \alpha_n$. Esta suma puede expresarse como un producto escalar entre el vector $\boldsymbol{a}$ y el arreglo conformado por los valores de α_n (ver ecuación 11).

$$\sum_{n=1}^{N} a_n \alpha_n = \begin{pmatrix} a_1 & \cdots & a_N \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \cdot \\ \alpha_N \end{pmatrix} = \boldsymbol{a}^\top \boldsymbol{\alpha} \tag{11}$$

Donde $\boldsymbol{\alpha} = (\alpha_1 \quad \cdots \quad \alpha_N)^\top$. Nótese que $\alpha_1 = a_1 t_1^2 k(x_1, x_1) + \cdots + a_N t_1 t_N k(x_1, x_N)$ y similarmente $\alpha_N = a_1 t_N t_1 k(x_N, x_1) + \cdots + a_N t_N^2 k(x_N, x_N)$, por tanto, $\boldsymbol{\alpha} = (\boldsymbol{t}\boldsymbol{t}^\top \odot \boldsymbol{K_{xx}})\boldsymbol{a}$, es decir (ecuación 12),

$$\sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) = \boldsymbol{a}^\top (\boldsymbol{t}\boldsymbol{t}^\top \odot \boldsymbol{K_{xx}})\boldsymbol{a} \tag{12}$$

Ahora se analiza el término $\sum_{n=1}^{N} a_n$ de la ecuación 1. Se presenta un producto escalar similar al del caso descrito en la ecuación 11, sin embargo, en este escenario, el producto está asociado entre $\boldsymbol{a}$ y un vector de unos (ver ecuación 13)

$$\sum_{n=1}^{N} a_n = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ \cdot \\ a_N \end{pmatrix} = \mathbf{1}^\top \boldsymbol{a} \tag{13}$$

Donde $\mathbf{1}^\top = (1 \quad \cdots \quad 1)$ donde $\mathbf{1} \in \mathbb{R}^{\boldsymbol{N}}$. Nótese que las N restricciones $0 \leq a_n \leq C$ podrán ser expresadas de forma vectorial, tal como se enseña en la ecuación 14.

$$\begin{pmatrix} 0 \\ \cdot \\ 0 \end{pmatrix} \leq \begin{pmatrix} a_1 \\ \cdot \\ a_N \end{pmatrix} \leq \begin{pmatrix} C \\ \cdot \\ C \end{pmatrix}$$
$$\mathbf{0} \preceq \boldsymbol{a} \preceq \boldsymbol{C} \tag{14}$$

Con $\mathbf{0} = (0 \quad \cdots \quad 0)^\top$, $\mathbf{0} \in \mathbb{R}^{\boldsymbol{N}}$ y $\boldsymbol{C} = (C \quad \cdots \quad C)^\top$, $\boldsymbol{C} \in \mathbb{R}^{\boldsymbol{N}}$. Para completar el análisis, consideremos el término $\sum_{n=1}^{N} a_n t_n$. Su expresión vectorial equivalente será un producto escalar entre los vectores $\boldsymbol{t}^\top$ y $\boldsymbol{a}$, tal como se muestra en la ecuación 15.

$$\sum_{n=1}^{N} a_n t_n = \begin{pmatrix} t_1 & \cdot & t_N \end{pmatrix} \begin{pmatrix} a_1 \\ \cdot \\ a_N \end{pmatrix} = \boldsymbol{t}^\top \boldsymbol{a} \tag{15}$$

Por lo tanto, al reemplazar las ecuaciones 12, 13, 14 y 15 en las ecuaciones 1 y 2, se obtiene la representación vectorial del problema descrito en la ecuación 2 (ver ecuación 16).

$$\max_{\boldsymbol{a}} \tilde{L}(\boldsymbol{a}) = \max_{\boldsymbol{a}} \left(\mathbf{1}^\top \boldsymbol{a} \ -\frac{1}{2} \boldsymbol{a}^\top (\boldsymbol{t}\boldsymbol{t}^\top \odot \boldsymbol{K_{xx}})\boldsymbol{a} \right)$$
$$sujeto\ a:$$
$$\mathbf{0} \preceq \boldsymbol{a} \preceq \boldsymbol{C} \tag{16}$$

$$\boldsymbol{t}^{\top}\boldsymbol{a} \quad = 0$$

Notar que el análisis realizado anteriormente puede ser extendido para la predicción (dada en la ecuación 3) cuando se tiene una matriz $\boldsymbol{z} = (z_1 \quad \cdots \quad z_r)^{\top}$, $\boldsymbol{z} \in \mathbb{R}^{r \times j}$ como entrada. Siguiendo un desarrollado similar al presentado en la ecuación 8, la ecuación 3 puede ser expresada en formato vectorial en función de $\boldsymbol{z}$, tal como se enseña en la ecuación 17. Donde $\boldsymbol{K_{zx}}$ es la matriz kernel entre las matrices $\boldsymbol{z}$ y $\boldsymbol{x}$. Es importante aclarar que la predicción en este caso tendrá la estructura $y(\boldsymbol{z}) = (y(z_1) \quad \cdots \quad y(z_r))^{\top}$, $y(\boldsymbol{z}) \in \mathbb{R}^{\boldsymbol{r}}$.

Con

$$y(\boldsymbol{z}) = sign\{(\boldsymbol{a} \odot \boldsymbol{t})^{\top}(\boldsymbol{K_{zx}})^{\top} + b\}$$
$$b = \frac{1}{N}(\boldsymbol{t}^{\top} - (\boldsymbol{a} \odot \mathbf{1})^{\top}\boldsymbol{K_{xx}})\mathbf{1} \tag{17}$$

Del análisis anterior se identifican cuatro operaciones básicas que permiten transformar la ecuación 1 en su forma vectorial (ecuación 16). La primera, es el producto externo del vector de etiquetas $\boldsymbol{t}$, que reúne todos los productos $t_n t_m$. En segundo lugar, el producto Hadamard, que asocia cada uno de los valores $t_n t_m$ con su respectivo kernel $k(x_n, x_m)$, en tercer lugar, la multiplicación entre una matriz y un vector, cuyo resultado es un vector. Finalmente, un producto escalar que permite expresar una suma como un producto interno entre dos vectores. Estas son, en síntesis, las operaciones que permiten llevar la ecuación 1 a su equivalente vectorial.

2.3. La SVM para clasificación como un QP

Un QP presenta la estructura descrita en la ecuación 18. Donde $\boldsymbol{u}^{\top} = (u_1 \quad \cdots \quad u_N)$, con $\boldsymbol{P}_{N\times N}$, $\boldsymbol{q}_{1\times N}$, $\boldsymbol{G}_{M\times N}$, $\boldsymbol{h}_{M\times 1}$, $\boldsymbol{lb}_{N\times 1}$, $\boldsymbol{ub}_{N\times 1}$ y $\boldsymbol{A}_{K\times N}$ [26]. El valor de M define el número de inecuaciones en el término $\boldsymbol{Gu} \leq \boldsymbol{h}$ mientras que N fija el total de desigualdades en el término $\boldsymbol{lb} \leq \boldsymbol{u} \leq \boldsymbol{ub}$, finalmente K establece el total de igualdades en relación a igualdad $\boldsymbol{Au} = \mathbf{0}_{K\times 1}$. Es importante anotar que minimizar una función es equivalente a maximizar el negativo de tal función [11].

$$\min_{\boldsymbol{u}} \frac{1}{2}\boldsymbol{u}^{\top}\boldsymbol{P}\boldsymbol{u} + \boldsymbol{q}^{\top}\boldsymbol{u} = \max_{\boldsymbol{u}} -\frac{1}{2}\boldsymbol{u}^{\top}\boldsymbol{P}\boldsymbol{u} - \boldsymbol{q}^{\top}\boldsymbol{u}$$
$$sujeto\ a:$$
$$\boldsymbol{Gu} \leq \boldsymbol{h} \tag{18}$$
$$\boldsymbol{lb} \leq \boldsymbol{u} \leq \boldsymbol{ub}$$
$$\boldsymbol{Au} = \boldsymbol{B}_{K\times 1}$$

El problema enseñado en la ecuación 16 puede ser expresado siguiendo el formato enseñado en la ecuación 18, sólo es necesario establecer las equivalencias, tal como se resume en las igualdades dadas en la ecuación 19. Donde $K = 1$ y que el término $\boldsymbol{Gu} \leq \boldsymbol{h}$ no aplica en el problema presentado en la ecuación 16 (para ver un caso donde esta condición aplica ver ejemplos sección 2.7 modelo V-SVM para clasificación).

$$\boldsymbol{u} = \boldsymbol{a}$$
$$\boldsymbol{q}^{\top} = -\mathbf{1}^{\top}$$
$$\boldsymbol{P} = \boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K_{xx}} \tag{19}$$
$$\boldsymbol{lb} = \mathbf{0}$$
$$\boldsymbol{ub} = \boldsymbol{C}$$
$$\boldsymbol{A} = \boldsymbol{t}^{\top}$$
$$\boldsymbol{B} = 0$$

Se dice que el QP expuesto en la ecuación 18 es convexo si la matriz $\boldsymbol{P}$ es semidefinida positiva. Una condición suficiente para que esto ocurra es que la matriz $\boldsymbol{K_{xx}}$ sea también semidefinida positiva. La convexidad del QP garantiza la existencia de una solución óptima global al problema presentado en 18, sin embargo, esto no implica la unicidad en la solución (pueden existir soluciones diferentes del vector $\boldsymbol{a}$ que conduzcan al mismo mínimo global). Esto se debe a que no puede asegurarse que el QP sea estrictamente convexo, es decir, que la matriz $\boldsymbol{P}$ sea una matriz definida positiva [27, 28]. En la práctica, la convexidad

queda garantizada seleccionado un kernel que sea semidefinidos positivos, como el kernel lineal, polinomial o de base radial [4].

2.4. El algoritmo R-SVM como un QP

Un modelo de regresión utilizando una SVM también puede formulase de manera similar a la ecuación 1. En este contexto, la función Lagrangiana dual para el algoritmo R-SVM es presentada en la ecuación 20 [4].

$$\tilde{L}(\boldsymbol{a},\hat{\boldsymbol{a}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(x_n,x_m) - \varphi\sum_{n=1}^{N}(a_n + \hat{a}_n) + \sum_{n=1}^{N}(a_n - \hat{a}_n)t_n$$

$$0 \le a_n \le C \qquad (20)$$

$$0 \le \hat{a}_n \le C$$

$$\sum_{n=1}^{N} a_n - \hat{a}_n = 0$$

Con $\hat{\boldsymbol{a}} = (\hat{a}_1 \quad \hat{a}_2 \quad \cdots \quad \hat{a}_N)^\top$, $\hat{\boldsymbol{a}} \in \mathbb{R}^N$ donde φ es la constante de sensibilidad [4] y t_n son etiquetas tal que $t_n \in \mathbb{R}$. Los vectores $\boldsymbol{a}$ y $\hat{\boldsymbol{a}}$ son los multiplicadores de Lagrange asociados a la función $\tilde{L}$. Nótese que, en este caso, la función $\tilde{L}$ se encuentra en términos de $2N$ multiplicadores de Lagrange, asimismo se tiene $4N$ restricciones asociadas a las desigualdades $0 \le a_n \le C$ y $0 \le \hat{a}_n \le C$. Para determinar los vectores $\boldsymbol{a}$ y $\hat{\boldsymbol{a}}$ se utiliza el mismo principio expuesto en la ecuación 2. Es decir, la idea es computar los vectores $\boldsymbol{a}$ y $\hat{\boldsymbol{a}}$ que maximizan la función $\tilde{L}$ y cuyos valores cumplan con las restricciones expresadas en la ecuación 20. Para ello, es necesario expresar la ecuación 20 en forma vectorial, lo que permitirá relacionarlo con un QP (ver ecuación 18). La predicción (para el modelo mostrado en la ecuación 20) para una entrada z puede realizarse a partir de la ecuación 21.

$$y(z) = \sum_{n=1}^{N}(a_n - \hat{a}_n)k(z,x_n) + b \qquad (21)$$

Con

$$b = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - \varphi - \sum_{m=1}^{N}(a_m - \hat{a}_m)k(x_n,x_m)\right)$$

A continuación, se examina cada uno de los términos de la ecuación 20, comenzando con el análisis del término $\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(x_n,x_m)$. Si se realiza el cambio de variable $\delta_n = (a_n - \hat{a}_n)$ y $\delta_m = (a_m - \hat{a}_m)$, esta toma la forma $\sum_{n=1}^{N}\sum_{m=1}^{N}\delta_n 1 \delta_m 1 k(x_n,x_m)$. Notamos que la estructura es similar a la presentada en la ecuación 6, donde se ha establecido (por analogía) que $t_n = t_m = 1$. Por otro lado, se debe tener cuidado dado que δ_n y δ_m estarán asociados a dos multiplicadores de Lagrange distintos para cada valor de n y m (ver ecuación 22).

$$\sum_{n=1}^{N}\sum_{m=1}^{N}\delta_n\delta_m k(x_n,x_m) = (\delta_1 \quad \cdots \quad \delta_N)\begin{pmatrix}1\\ \cdot \\ 1\end{pmatrix}(1 \quad \cdots \quad 1) \odot \begin{pmatrix}k(x_1,x_1) & \cdot & k(x_1,x_N)\\ \cdot & \cdot & \cdot \\ k(x_N,x_1) & \cdot & k(x_N,x_N)\end{pmatrix}\begin{pmatrix}\delta_1\\ \cdot \\ \delta_N\end{pmatrix} \qquad (22)$$

$$= (a_1 - \hat{a}_1 \quad \cdots \quad a_N - \hat{a}_N)\begin{pmatrix}1\\ \cdot \\ 1\end{pmatrix}(1 \quad \cdots \quad 1) \odot \begin{pmatrix}k(x_1,x_1) & \cdot & k(x_1,x_N)\\ \cdot & \cdot & \cdot \\ k(x_N,x_1) & \cdot & k(x_N,x_N)\end{pmatrix}\begin{pmatrix}a_1 - \hat{a}_1\\ \cdot \\ a_N - \hat{a}_N\end{pmatrix}$$

Notar que se ha utilizado nuevamente las operaciones vectoriales y matriciales identificadas en el caso del algoritmo C-SVM. La ecuación 22 puede modificarse de manera que todos los multiplicadores de Lagrange se puedan expresar como un vector, tal como se enseña en la ecuación 23. Es de aclarar que se ha utilizado el hecho de que $a_1 - \hat{a}_1 + \cdots + a_N - \hat{a}_N = a_1 + \cdots + a_N - \hat{a}_1 - \cdots - \hat{a}_N$.

$$= (a_1 - \hat{a}_1 \quad \cdots \quad a_N - \hat{a}_N)\begin{pmatrix}1\\ \cdot \\ 1\end{pmatrix}(1 \quad \cdots \quad 1) \odot \begin{pmatrix}k(x_1,x_1) & \cdot & k(x_1,x_N)\\ \cdot & \cdot & \cdot \\ k(x_N,x_1) & \cdot & k(x_N,x_N)\end{pmatrix}\begin{pmatrix}a_1 - \hat{a}_1\\ \cdot \\ a_N - \hat{a}_N\end{pmatrix} =$$

$$= (a_1 \quad \cdots \quad a_N \quad \hat{a}_1 \quad \cdots \quad \hat{a}_N)\begin{pmatrix}1\\ \cdot \\ 1\\ -1\\ \cdot \\ -1\end{pmatrix}(1 \quad \cdots \quad 1 \quad -1 \quad \cdots \quad -1) \odot \boldsymbol{S}\begin{pmatrix}a_1\\ \cdot \\ a_N\\ \hat{a}_1\\ \cdot \\ \hat{a}_N\end{pmatrix} \tag{23}$$

$$\sum_{n=1}^{N}\sum_{m=1}^{N}\delta_n\delta_m k(x_n,x_m) = \ddot{\boldsymbol{a}}^{\top}[(\mathbf{1} \quad -\mathbf{1})(\mathbf{1} \quad -\mathbf{1})^{\top} \odot \boldsymbol{S}]\ddot{\boldsymbol{a}}$$

Con

$$\boldsymbol{S} = \begin{pmatrix}\boldsymbol{K_{xx}} & \boldsymbol{K_{xx}}\\ \boldsymbol{K_{xx}} & \boldsymbol{K_{xx}}\end{pmatrix}$$

Donde se ha definido $\ddot{\boldsymbol{a}} = (a_1 \quad \cdots \quad a_N \quad \hat{a}_1 \quad \cdots \quad \hat{a}_N)^{\top} = (\boldsymbol{a}, \quad \hat{\boldsymbol{a}})^{\top}, \ddot{\boldsymbol{a}} \in \mathbb{R}^{2N}$ y $(\mathbf{1} \quad -\mathbf{1}) = (1 \quad \cdots \quad 1 \quad -1 \quad \cdots \quad -1)^{\top} \in \mathbb{R}^{2N}$. Es de aclarar que los multiplicadores de Lagrange ahora se concentran en el vector $\ddot{\boldsymbol{a}}$. Por otro lado, la matriz $\boldsymbol{S} \in \mathbb{R}^{2N\times 2N}$, lo anterior se debe a que el producto escalar entre $(\mathbf{1} \quad -\mathbf{1})(\mathbf{1} \quad -\mathbf{1})^{\top} \in \mathbb{R}^{2N\times 2N}$. Ahora, se analiza el último término, este puede rescribirse como se presenta en la ecuación 24.

$$-\varphi\sum_{n=1}^{N}(a_n + \hat{a}_n) + \sum_{n=1}^{N}(a_n - \hat{a}_n)t_n = \sum_{n=1}^{N}([t_n - \varphi\,]a_n - [t_n + \varphi\,]\hat{a}_n) \tag{24}$$

La expresión 24 puede ser expresada aplicando el producto escalar para el vector $\ddot{\boldsymbol{a}}$ y la etiqueta t_n. En la ecuación 25 se desarrolla el procedimiento.

$$\sum_{n=1}^{N}([t_n - \varphi\,]a_n - [t_n + \varphi\,]\hat{a}_n) = (t_1 - \varphi \quad \cdots \quad t_N - \varphi \quad -(t_1 + \varphi) \quad \cdots \quad -(t_N + \varphi))\begin{pmatrix}a_1\\ \cdot \\ a_N\\ \hat{a}_1\\ \cdot \\ \hat{a}_N\end{pmatrix} \tag{25}$$

$$\sum_{n=1}^{N}([t_n - \varphi\,]a_n - [t_n + \varphi\,]\hat{a}_n) = (\boldsymbol{\theta}_1^{\top} \quad -\boldsymbol{\theta}_2^{\top})\ddot{\boldsymbol{a}}$$

Donde $\boldsymbol{\theta}_1 = (t_1 - \varphi \quad \cdots \quad t_N - \varphi)^{\top}$, $\boldsymbol{\theta}_1 \in \mathbb{R}^N$ y $\boldsymbol{\theta}_2 = (t_1 + \varphi \quad \cdots \quad t_N + \varphi)^{\top}, \boldsymbol{\theta}_2 \in \mathbb{R}^N$. Ahora, las $4N$ desigualdades puede ser expresadas vectorialmente, tal como se enseña en la ecuación 26.

$$\begin{pmatrix}0\\0\\\cdot\\\cdot\\0\end{pmatrix} \leq \begin{pmatrix}a_1\\\cdot\\a_N\\\hat{a}_1\\\cdot\\\hat{a}_N\end{pmatrix} \leq \begin{pmatrix}C\\C\\\cdot\\\cdot\\C\end{pmatrix} = \tag{26}$$

$$\begin{pmatrix}\mathbf{0}\\\mathbf{0}\end{pmatrix} \leq \ddot{\boldsymbol{a}} \leq \begin{pmatrix}\boldsymbol{C}\\\boldsymbol{C}\end{pmatrix}$$

Donde $\begin{pmatrix}\mathbf{0}\\\mathbf{0}\end{pmatrix} = (\mathbf{0} \quad \mathbf{0})^\top$ y $\begin{pmatrix}\boldsymbol{C}\\\boldsymbol{C}\end{pmatrix} = (\boldsymbol{C} \quad \boldsymbol{C})^\top$. Finalmente, el término $\sum_{n=1}^{N} a_n - \hat{a}_n = a_1 - \hat{a}_1 + \cdots + a_N - \hat{a}_N = a_1 + \cdots + a_N - \hat{a}_1 - \cdots - \hat{a}_N = 0$ puede expresarse como un producto escalar entre $(\mathbf{1} \quad -\mathbf{1})^\top$ y $\ddot{\boldsymbol{a}}$, como se presenta en la ecuación 27.

$$\sum_{n=1}^{N} a_n - \hat{a}_n = 0 = (1 \quad \cdots \quad 1 \quad -1 \quad \cdots \quad -1)\begin{pmatrix}a_1\\\cdot\\a_N\\\hat{a}_1\\\cdot\\\hat{a}_N\end{pmatrix} = (\mathbf{1} \quad -\mathbf{1})^\top \ddot{\boldsymbol{a}} = 0 \tag{27}$$

Al reemplazar las ecuaciones 23, 25, 26 y 27 en la ecuación 20, obtenemos la ecuación 28. A diferencia del problema plateado en la ecuación 16, la solución al problema de optimización presentado en la ecuación 27 devuelve el vector $\ddot{\boldsymbol{a}}$.

$$\max_{\ddot{\boldsymbol{a}}} \tilde{L}(\ddot{\boldsymbol{a}}) = \max_{\ddot{\boldsymbol{a}}} \left(-\frac{1}{2}\ddot{\boldsymbol{a}}^\top[(\mathbf{1} \quad -\mathbf{1})(\mathbf{1} \quad -\mathbf{1})^\top \odot \boldsymbol{S}]\ddot{\boldsymbol{a}} + (\boldsymbol{\theta}_1^\top \quad -\boldsymbol{\theta}_2^\top)\ddot{\boldsymbol{a}}\right)$$

$$sujeto\ a\text{:}$$

$$\begin{pmatrix}\mathbf{0}\\\mathbf{0}\end{pmatrix} \leq \ddot{\boldsymbol{a}} \leq \begin{pmatrix}\boldsymbol{C}\\\boldsymbol{C}\end{pmatrix} \tag{28}$$

$$(\mathbf{1} \quad -\mathbf{1})^\top \ddot{\boldsymbol{a}} = 0$$

Nótese que la ecuación 28 corresponde a un QP y presenta las equivalencias presentadas en la ecuación 29. En esta situación el término $\boldsymbol{Gu} \leq \boldsymbol{h}$ no presenta aplicación al problema presentado en la ecuación 28 (ver algoritmo V-SVM para regresión ejemplos sección 2.7).

$$\begin{gathered}\boldsymbol{u} = \ddot{\boldsymbol{a}}\\ \boldsymbol{q}^T = -(\boldsymbol{\theta}_1^\top \quad -\boldsymbol{\theta}_2^\top) = (-\boldsymbol{\theta}_1^\top \quad \boldsymbol{\theta}_2^\top)\\ \boldsymbol{P} = (\mathbf{1} \quad -\mathbf{1})(\mathbf{1} \quad -\mathbf{1})^\top \odot \boldsymbol{S}\\ \boldsymbol{lb} = \begin{pmatrix}\mathbf{0}\\\mathbf{0}\end{pmatrix}\\ \boldsymbol{ub} = \begin{pmatrix}\boldsymbol{C}\\\boldsymbol{C}\end{pmatrix}\\ \boldsymbol{A} = (\mathbf{1} \quad -\mathbf{1})^\top\\ \boldsymbol{b} = 0\end{gathered} \tag{29}$$

Para realizar una predicción utilizando la ecuación 21 sobre un vector de entrada $\boldsymbol{z}$, esta puede ser reformulada en una notación vectorial, como se presenta en la ecuación 30.

$$y(\boldsymbol{z}) = \left(\ddot{\boldsymbol{a}} \odot (\mathbf{1} \quad -\mathbf{1})\right)^{\top} (\boldsymbol{K_{zx}} \quad \boldsymbol{K_{zx}})^{\top} + b$$

Con

$$b = \frac{1}{N}\left(\boldsymbol{\theta}_1^{\top} - \left(\ddot{\boldsymbol{a}} \odot (\mathbf{1} \quad -\mathbf{1})\right)^{\top} \begin{pmatrix} \boldsymbol{K_{xx}} \\ \boldsymbol{K_{xx}} \end{pmatrix}\right)\mathbf{1} \tag{30}$$

Las condiciones KKT para el problema presentando en la ecuación 20 se resumen en la ecuación 31. En esta formulación, ξ_n y $\hat{\xi}_n$ son las variables de holgura (traducción del inglés 'slack variables'), que son necesarias para suavizar las restricciones en el problema de optimización de la ecuación 28 [4]. De manera similar al caso de clasificación, los vectores de soporte serán las soluciones de a_n y $\hat{a}_n$ distintos de cero.

$$\begin{gathered} a_n(\varphi + \xi_n + y(x_n) - t_n) = 0 \\ \hat{a}_n(\varphi + \hat{\xi}_n - y(x_n) + t_n) = 0 \\ (C - a_n)\xi_n = 0 \\ (C - \hat{a}_n)\hat{\xi}_n = 0 \end{gathered} \tag{31}$$

2.5. **El kernel en la implementación de la SVM**

En este trabajo la ecuación 32 define la representación matemática de un kernel o función de base radial (RBF) entre los vectores x_n y x_m. La operación $\|x_n - x_m\|$ indica norma euclidiana de la diferencia $x_n - x_m$. El valor de $k(x_n, x_m)$ será un escalar y en el caso en que $x_n = x_m$ se cumplirá que $k(x_n, x_m) = 1$. Note además que $k(x_n, x_m) = k(x_m, x_n)$. Esto implica que la matriz $\boldsymbol{K_{xx}}$ sea simétrica ($\boldsymbol{K_{xx}} = \boldsymbol{K_{xx}^{\top}}$), sin embargo, esta propiedad no se cumple para la matriz $\boldsymbol{K_{zx}}$ ($\boldsymbol{K_{zx}} \neq \boldsymbol{K_{zx}^{\top}}$). El parámetro $\frac{1}{2\sigma^2}$ conocido como la escala de longitud (length scale en inglés), controla la sensibilidad, así como el alcance de la RBF sobre los vectores de entrada x_n y x_m [29]. En este trabajo se utilizará la RBF para determinar las matrices $\boldsymbol{K_{xx}}$ y $\boldsymbol{K_{zx}}$, las cuales son imprescindibles en la solución de los problemas presentados en las ecuaciones 16 y 28.

$$k(x_n, x_m) = \emptyset(x_n)\emptyset(x_m^{\top}) = exp\left(-\frac{1}{2\sigma^2}\|x_n - x_m\|^2\right) \tag{32}$$

Es de resaltar que la matriz $\boldsymbol{K_{xx}}$, al ser construida mediante un kernel de la forma presentada en la ecuación 32, garantiza sea al menos semidefinida positiva [4]. Esto implica que los QP expuestos en las ecuaciones 19 y 28 sean convexos.

2.6. **Plano de separación: indicador visual de rendimiento de la SVM en clasificación**

Para analizar cómo la SVM clasifica los datos, es crucial determinar el plano que separa las clases en un problema de clasificación binaria (también se le conoce como frontera de decisión o curva de decisión). El plano de separación viene dado por la curva de nivel $y(z) = 0$, esto implica que (ver ecuación 33).

$$0 = \sum_{n=1}^{N} a_n t_n k(z, x_n) + b \tag{33}$$

La ecuación 33 muestra cómo la estructura del término $k(z, x_n)$ influye en el comportamiento de la frontera de decisión, permitiendo curvas de nivel, tanto lineales y no lineales. Esto le proporciona a la SVM la flexibilidad necesaria para determinar una separación adecuada de las clases. Infortunadamente, el plano de decisión sólo podrá observarse para $N \leq 3$.

2.7. **Ejemplo de QP en diferentes versiones del algoritmo SVM**

En esta sección, se presentan las formulaciones QP para algunos algoritmos de SVM documentados en la literatura. En todos los casos, se parte de la función Lagrangiana en su versión dual en notación sigma y luego se presenta su versión matricial.

- **V-SMV para clasificación**

En la ecuación 34 se presenta la forma dual del algoritmo V-SVM. Donde ν es un parámetro del algoritmo [15].

$$\tilde{L}(\boldsymbol{a}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m)$$

$$sujeto\ a:$$

$$0 \leq a_n \leq \frac{1}{N} \tag{34}$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

$$\sum_{n=1}^{N} a_n \geq \nu$$

Su forma matricial se enseña en la ecuación 35.

$$\tilde{L}(\boldsymbol{a}) = -\frac{1}{2}\boldsymbol{a}^{\top}(\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}})\boldsymbol{a}$$

$$sujeto\ a:$$

$$\boldsymbol{0} \preceq \boldsymbol{a} \preceq \frac{\boldsymbol{1}}{N} \tag{35}$$

$$\boldsymbol{t}^{\top}\boldsymbol{a} \quad = 0$$

$$-\boldsymbol{1}^{\top}\boldsymbol{a} \leq -\nu$$

- **V-SVM para regresión**

En las ecuaciones 36 y 37 se presentan la forma dual y matricial respectivamente, para el algoritmo V-SVM en regresión, donde ν y C_1 son parámetros del algoritmo [15].

$$\tilde{L}(\boldsymbol{a}, \widehat{\boldsymbol{a}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(x_n, x_m) + \sum_{n=1}^{N}(a_n - \hat{a}_n)t_n$$

$$sujeto\ a:$$

$$0 \leq a_n \leq \frac{C}{N} \tag{36}$$

$$0 \leq \hat{a}_n \leq \frac{C}{N}$$

$$\sum_{n=1}^{N} a_n - \hat{a}_n = 0$$

$$\sum_{n=1}^{N} a_n + \hat{a}_n \leq \nu\, C$$

$$\tilde{L}(\boldsymbol{a}) = -\frac{1}{2}\ddot{\boldsymbol{a}}^{\top}[(\mathbf{1} \quad -\mathbf{1})(\mathbf{1} \quad -\mathbf{1})^{\top} \odot \boldsymbol{S}]\ddot{\boldsymbol{a}} + (\boldsymbol{t}^{\top} \quad -\boldsymbol{t}^{\top})^{\top}\ddot{\boldsymbol{a}}$$

$$sujeto\ a\text{:}$$

$$\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\end{pmatrix} \leq \ \ddot{\boldsymbol{a}} \leq \frac{1}{N}\begin{pmatrix}\boldsymbol{C}\\ \boldsymbol{C}\end{pmatrix} \tag{37}$$

$$(\mathbf{1} \quad -\mathbf{1})^{\top}\ddot{\boldsymbol{a}} = 0$$

$$(\mathbf{1} \quad \mathbf{1})^{\top}\ddot{\boldsymbol{a}} \leq \nu C$$

- **B-SVM**

En las ecuaciones 38 y 39 se presentan la forma dual y matricial respectivamente, para el algoritmo B-SVM en regresión. El significado de los parámetros C_1 y C_2 puede consultarse en [14].

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) - \frac{1}{4C_1}\sum_{n=1\{t_n=1\}}^{N} a_n^2 - \frac{1}{4C_2}\sum_{n=1\{t_n=-1\}}^{N} a_n^2$$

$$sujeto\ a\text{:}$$

$$0 \leq a_n \leq C_1 \quad si\ t_n = 1 \tag{38}$$

$$0 \leq a_n \leq C_2 \quad si\ t_n = -1$$

$$\tilde{L}(\boldsymbol{a}) = -\frac{1}{2}\boldsymbol{a}^{\top}(\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K}_{\boldsymbol{xx}} + \boldsymbol{D})\boldsymbol{a} + \mathbf{1}^{\top}\boldsymbol{a}$$

$$sujeto\ a\text{:}$$

$$\mathbf{0} \leq \boldsymbol{a} \leq \boldsymbol{C}_T$$

Con

(39)

$$\boldsymbol{D} = \begin{pmatrix} D_1 & 0 & \cdots & 0 \\ 0 & D_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_N \end{pmatrix}$$

$$\boldsymbol{C}_T = (C_{T_1} \quad \cdots \quad C_{T_N})^{\top}$$

Donde

$$D_n = \begin{cases} \dfrac{1}{2C_1} \ si\ t_n = 1 \\ \dfrac{1}{2C_2} \ si\ t_n = -1 \end{cases}$$

- **CS-SVM**

En las ecuaciones 40 y 41 se presentan la forma dual y matricial respectivamente, para el algoritmo B-SVM en regresión. Información detallada del funcionamiento de este algoritmo puede consultarse [17]

$$\tilde{L}(\boldsymbol{a}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) + \sum_{n=1\{t_n=1\}}^{N} a_n + \vartheta \sum_{n=1\{t_n=-1\}}^{N} a_n$$

$$sujeto\ a:$$

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{40}$$

$$0 \le a_n \le CC_1 \quad si\ t_n = 1$$

$$0 \le a_n \le \frac{C_1}{\vartheta} \quad si\ t_n = -1$$

$$\tilde{L}(\boldsymbol{a}) = -\frac{1}{2}\boldsymbol{a}^{\top}(\boldsymbol{t}\boldsymbol{t}^{\top} \odot \boldsymbol{K_{xx}})\boldsymbol{a} + \boldsymbol{L}^{\top}\boldsymbol{a}$$

$$sujeto\ a:$$

$$\boldsymbol{t}^{\top}\boldsymbol{a} \quad = 0$$

$$\boldsymbol{0} \le \boldsymbol{a} \le \boldsymbol{C}_T$$

Con

$$\boldsymbol{L} = (L_1 \quad \cdots \quad L_N)^{\top}$$

$$\boldsymbol{C}_T = (C_{T_1} \quad \cdots \quad C_{T_N})^{\top} \tag{41}$$

Donde

$$L_n = \begin{cases} 1 \ si\ t_n = 1 \\ \vartheta\ si\ t_n = -1 \end{cases}$$

$$C_{T_n} = \begin{cases} CC_1 \ si\ t_n = 1 \\ \frac{C_1}{\delta}\ si\ t_n = -1 \end{cases}$$

2.8. **Los datos en la implementación de la SVM**

En este trabajo se utilizó dos bases de datos para probar el funcionamiento de la SVM descrita en la ecuación 16. Estas contienen datos sintéticos con dos variables de entrada $x_i = (x_{i1}, x_{i2})^{\top}$ asociadas a sus respectivas etiquetas. En la Figura 1 se presenta un diagrama de dispersión de los datos, donde se puede identificar dos clases o etiquetas diferenciadas por color. El gráfico de dispersión izquierdo de la Figura 1 evidencia que será factible encontrar una curva de decisión que separa las dos clases, a diferencia del gráfico derecho de la Figura 1, donde hay un leve traslape entre algunos datos. Al problema de asignar un dato a una de dos clases se le conoce como clasificación binaria, que es el caso abordado utilizando los datos de la Figura 1.

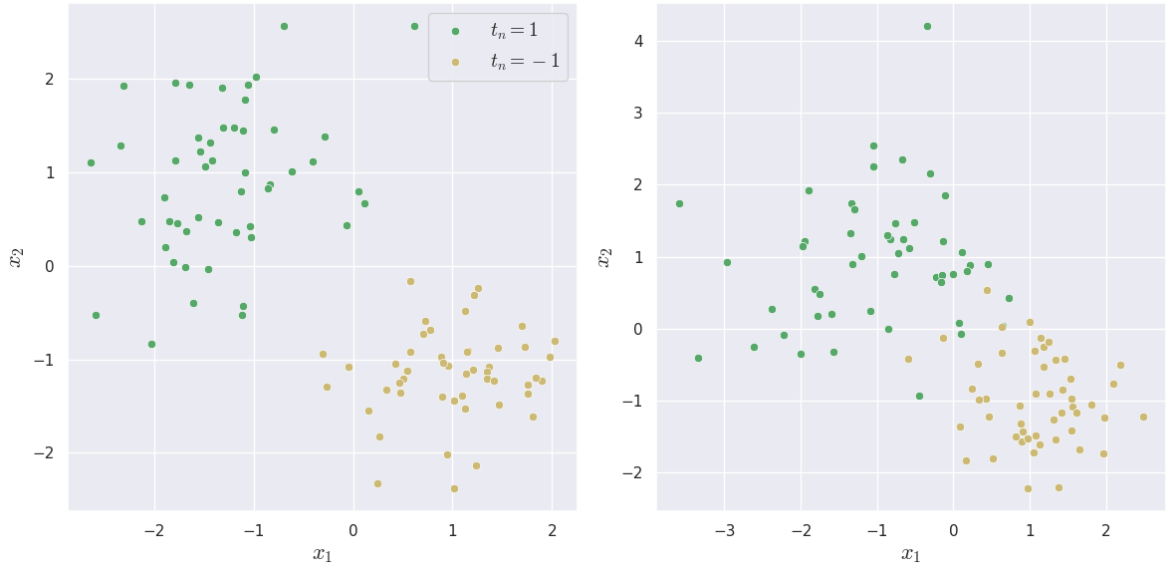

Figura 1: Diagramas de dispersión de las bases de datos para clasificación donde Clase 1 y Clase 2 indican $t_n = 1$ y $t_n = -1$ respectivamente. Por otro lado, los ejes x_1 y x_2 corresponden a las componentes del vector de entrada x_i. Figura izquierda base de datos clasificación 1. Figura derecha base de datos clasificación 2.

Similarmente para el problema presentado en la ecuación 26 se utilizará una base de datos compuesta de información sintética. En la Figura 2 se enseña el diagrama de dispersión entre la entrada y la salida.

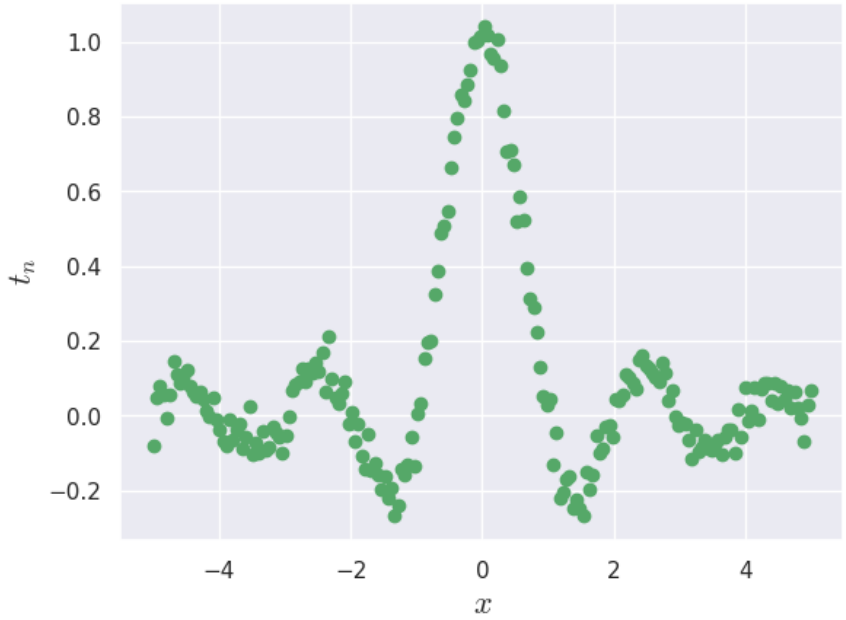

Figura 2: Diagrama de dispersión de la base de datos para regresión.

Para el proceso de entrenamiento, se utilizó el 70% de los datos de cada base de datos. Este porcentaje corresponde a los términos $\boldsymbol{x}$ y $\boldsymbol{t}$ utilizados para inferir los multiplicadores de Lagrange, según las ecuaciones 16 y 27. El 30% restante se reservó para validación del modelo, los cuales corresponden a los términos $\boldsymbol{z}$ y $\boldsymbol{y} = (y_1 \quad \cdots \quad y_r)^\top$ (salida real). Es importante destacar que la selección de los datos de entrenamiento y validación se realizó por medio de permutaciones aleatorias.

2.9. **Evaluación del rendimiento de SVM mediante métricas de validación**

Para evaluar el desempeño de la SVM tanto para la tarea de clasificación y regresión se utilizó las métricas definidas en la Tabla 2 [30, 31, 32].

Tabla 2: Métricas de desempeño en clasificación y regresión

Métrica	Definición	Tarea
Especificidad TNR	$TNR = \frac{VN}{VN + FP}$	Clasificación
Sensibilidad TPR	$TPR = \frac{VP}{VP + FN}$	Clasificación
Precisión PPV	$PPV = \frac{VP}{VP + FP}$	Clasificación
Exactitud Acc	$Acc = \frac{VP + VN}{VP + VN + FP + FN}$	Clasificación
Error Cuadrático Medio $ERMS$	$ERMS_{y(\boldsymbol{z})\boldsymbol{y}} = \sqrt{\frac{(y(\boldsymbol{z}) - \boldsymbol{y})^{\top}(y(\boldsymbol{z}) - \boldsymbol{y})}{r}}$	Regresión
Coeficiente de correlación de pearson ρ	$\rho_{y(\boldsymbol{z})\boldsymbol{y}} = \frac{Cov(\boldsymbol{y}, y(\boldsymbol{z}))}{\sigma_{y(\boldsymbol{z})}\sigma_{\boldsymbol{y}}}$ $Cov(\boldsymbol{y}, y(\boldsymbol{z})) = \frac{1}{r}\sum_{n=1}^{r}(y_n - \bar{\boldsymbol{y}})(y(z)_n - \overline{y(\boldsymbol{z})})$	Regresión

Donde VP, VN, FP y FN se refieren a los verdaderos positivos, los verdaderos negativos, los falsos positivos y falsos negativos, respectivamente. Ahora, un rendimiento adecuado de la SVM en clasificación busca que las métricas (TNR, TPR, PPV y Acc) estén lo más cercanas posible a la unidad. Por otro lado, el término $Cov(\boldsymbol{y}, y(\boldsymbol{z}))$ es la covarianza entre las muestras $\boldsymbol{y}$ y $y(\boldsymbol{z})$, $\sigma_{y(\boldsymbol{z})}$ y $\sigma_{\boldsymbol{y}}$ son las desviaciones estándar de $y(\boldsymbol{z})$ y $\boldsymbol{y}$, respectivamente, mientras que $\overline{y(\boldsymbol{z})}$ y $\bar{\boldsymbol{y}}$ son las medias aritméticas de $\boldsymbol{y}$ y $y(\boldsymbol{z})$ respectivamente. Un buen desempeño de la SVM en regresión debe arrojar un valor de ρ cercano a la unidad y un valor bajo de $ERMS$.

2.10. **Algoritmo para implementar la SVM en clasificación y regresión**

En la Figura 3 se presentan los algoritmos desarrollados para implementar una SVM en clasificación y regresión. Nótese que estos sintetizan los procedimientos descritos en el marco metodológico, comenzando con la selección de los datos, la construcción del QP y, por último, la evaluación de la SVM tanto para clasificación como para regresión. Es importante aclarar que la notación matemática $\boldsymbol{X}\backslash\boldsymbol{x}$ utilizada en los algoritmos de la Figura 3 indica datos de $\boldsymbol{X}$ que no pertenecen a $\boldsymbol{x}$.

Algoritmo para clasificación SVM

Entrada: Datos X, C

Salidas: $a_n, y(\mathbf{z})$

Inicio

// Definición datos de entrenamiento y validación

$N_X \leftarrow tamaño(X)$

$N \leftarrow 0.7N_x$

$[\mathbf{x}, \mathbf{t}] \leftarrow RandomSelect(X, N)$

$[\mathbf{z}, \mathbf{y}] \leftarrow X \backslash \mathbf{x}$

// Cálculo de matrices kernel $\mathbf{K}$

$\mathbf{K_{xx}} = RBF(\mathbf{x}, \mathbf{x})$

$\mathbf{K_{zx}} = RBF(\mathbf{z}, \mathbf{x})$

//Construcción del modelo de programación

// Cuadrático convexo

$\mathbf{P} \leftarrow \mathbf{t}\mathbf{t}^\top \odot \mathbf{K_{xx}}$

$\mathbf{q}^\top \leftarrow -\mathbf{1}^\top$

$\mathbf{lb} \leftarrow \mathbf{0}$

$\mathbf{ub} \leftarrow \mathbf{C}$

$\mathbf{A} \leftarrow \mathbf{t}^\top$

$\mathbf{B} \leftarrow 0$

$\mathbf{a} \leftarrow progquad(\mathbf{P}, \mathbf{q}^\top, \mathbf{lb}, \mathbf{ub}, \mathbf{A}, \mathbf{B})$

// Cálculo de la predicción en datos de validación

$b \leftarrow \frac{1}{N}(\mathbf{t}^\top - (\mathbf{a} \odot \mathbf{t})^\top \mathbf{K_{xx}})\mathbf{1}$

$y(\mathbf{z}) \leftarrow sign((\mathbf{a} \odot \mathbf{t})^\top \mathbf{K_{zx}}^\top + b)$

// Cálculo métricas de rendimiento

$[TRN, TPR, PVV, Acc] \leftarrow Metricas(y(\mathbf{z}), \mathbf{y})$

// Determinación curva de decisión

$CurvaNivel(\mathbf{a}, b, \mathbf{x}, \mathbf{t})$

Fin

Algoritmo para regresión con SVM

Entrada: Datos X, C y φ

Salidas: $\ddot{\mathbf{a}}, y(\mathbf{z})$

Inicio

// Definición datos de entrenamiento y validación

$N_X \leftarrow tamaño(X)$

$N \leftarrow 0.7N_x$

$[\mathbf{x}, \mathbf{t}] \leftarrow RandomSelect(X, N)$

$[\mathbf{z}, \mathbf{y}] \leftarrow X \backslash \mathbf{x}$

// Cálculo de matrices kernel $\mathbf{K}$

$\mathbf{K_{xx}} \leftarrow RBF(\mathbf{x}, \mathbf{x})$

$\mathbf{K_{zx}} \leftarrow RBF(\mathbf{z}, \mathbf{x})$

$\mathbf{S} \leftarrow \begin{pmatrix} \mathbf{K_{xx}} & \mathbf{K_{xx}} \\ \mathbf{K_{xx}} & \mathbf{K_{xx}} \end{pmatrix}$

//Construcción del modelo de programación

// Cuadrático convexo

$\mathbf{P} \leftarrow (\mathbf{1} \quad -\mathbf{1})(\mathbf{1} \quad -\mathbf{1})^\top \odot \mathbf{S}$

$\boldsymbol{\theta}_1 \leftarrow (t_1 - \varphi \quad \cdots \quad t_N - \varphi)^\top$

$\boldsymbol{\theta}_2 \leftarrow (t_1 + \varphi \quad \cdots \quad t_N + \varphi)^\top$

$\mathbf{q}^\top \leftarrow (-\boldsymbol{\theta}_1^\top \quad \boldsymbol{\theta}_2^\top)$

$\mathbf{lb} \leftarrow \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$

$\mathbf{ub} \leftarrow \begin{pmatrix} \mathbf{C} \\ \mathbf{C} \end{pmatrix}$

$\mathbf{A} \leftarrow (\mathbf{1} \quad -\mathbf{1})^\top$

$\mathbf{B} \leftarrow \mathbf{0}$

$\ddot{\mathbf{a}} \leftarrow progquad(\mathbf{P}, \mathbf{q}^\top, \mathbf{lb}, \mathbf{ub}, \mathbf{A}, \mathbf{B})$

// Cálculo de la predicción en datos de validación

$b \leftarrow \frac{1}{N}\left(\boldsymbol{\theta}_1^\top - (\ddot{\mathbf{a}} \odot (\mathbf{1} \quad -\mathbf{1}))^\top \begin{pmatrix} \mathbf{K_{xx}} \\ \mathbf{K_{xx}} \end{pmatrix}\right)\mathbf{1}$

$y(\mathbf{z}) \leftarrow (\ddot{\mathbf{a}} \odot (\mathbf{1} \quad -\mathbf{1}))^\top (\mathbf{K_{zx}} \quad \mathbf{K_{zx}})^\top + b$

// Cálculo métricas de rendimiento

$[ERMS_{y(\mathbf{z})\mathbf{y}}, \rho_{y(\mathbf{z})\mathbf{y}}] \leftarrow Metricas(y(\mathbf{z}), \mathbf{y})$

// Gráficos

$Graficos(\mathbf{x}, \mathbf{t}, \mathbf{z}, \mathbf{y}, y(\mathbf{z}), \ddot{\mathbf{a}})$

Fin

Figura 3: Algoritmos para implementar y validar la SVM en tareas de clasificación binaria y regresión.

2.11. Experimentos, herramientas para la comunicación de resultados y definición de parámetros

Las pruebas se concentraron en determinar las métricas presentadas en la Tabla 2. Además, se incluyen gráficos que ilustran diferentes aspectos acerca del rendimiento de la SVM. En la tarea de clasificación, se enseña el comportamiento de la curva de decisión una vez el vector $\boldsymbol{a}$ ha sido determinado. Este gráfico también incluye los vectores de soporte (los valores $a_n \neq 0$), lo que permite identificar las entradas x_n que aportan información crucial al modelo. Similarmente, en la SVM para regresión, se presenta un gráfico que

compara la predicción $y(\boldsymbol{z})$ con las etiquetas reales $\boldsymbol{y}$. En este gráfico también se enseñan los vectores de soporte. Finalmente, se incluye un gráfico $y(\boldsymbol{z})\ vs\ \boldsymbol{y}$ para observar la relación entre estas variables [32]. Por otro lado, en la Tabla 3 se definen los valores de los parámetros e hiper-parámetros utilizados en este trabajo, se aclara que la selección de estos valores no siguió ninguna técnica específica y que su único propósito es ilustrar su uso en la implementación de la SVM.

Tabla 3: Valores por defecto en los parámetros para implementar la SVM

Parámetro	Valor de referencia
C	10.00
φ	0.05
$\frac{1}{2\sigma^2}$	1.67

2.12. Herramientas computacionales para la solución del problema de programación cuadrática convexa

Se probaron dos herramientas computacionales para resolver los problemas descritos en las ecuaciones 16 y 28. El paquete *quadprog* de *MATLAB* y la librería *qpsolvers* para *Python* [33]. Ambos utilizan el método de punto interior convexo en el cómputo de la solución. Se utiliza la biblioteca *numpy* para las operaciones con vectores y matrices, así como las bibliotecas *matplotlib* y *seaborn* para la elaboración de gráficos en Python [34,35,36]. Los resultados incluirán las métricas descritas en la Tabla 2, esto con el propósito de proporcionar una medida comparativa en los resultados computados con ambas herramientas computacionales.

3. RESULTADOS Y DISCUSIÓN

En la Tabla 4 se reportan los resultados de las métricas de desempeño definidas en la Tabla 2 para los datos $y(\boldsymbol{z})$ y $\boldsymbol{y}$. Se observa un rendimiento muy similar para cada una de las herramientas de cómputo utilizadas. Se destaca el buen comportamiento de la SVM en la base de datos 1 clasificación y base de datos regresión. Sin embargo, hay un mejor rendimiento de la solución computada por *quadprog* en los valores de las métricas TNR y PPV para la base de datos clasificación 2.

Tabla 4: Reporte métricas de rendimiento

Base de datos 1 clasificación	*quadprog*	*qpsolvers*
TNR	*1*	*1*
TPR	*1*	*1*
PPV	*1*	*1*
Acc	*1*	*1*
Base de datos 2 clasificación		
TNR	***1***	*0.85*
TPR	*0.89*	*1*
PPV	***1***	*0.89*
Acc	0.93	*0.93*
Base de datos regresión		
$ERMS_{y(\boldsymbol{z})\boldsymbol{y}}$	*0.06*	*0.06*
$\rho_{y(\boldsymbol{z})\boldsymbol{y}}$	*0.99*	*0.99*

En la Figura 4, se ilustra la curva de decisión que divide cada una de las clases en un problema de clasificación binaria. Nótese como la separación entre los dos planos se realiza de forma no lineal, lo cual es importante, dado que rara vez los datos pueden ser separados de forma lineal. El gráfico de la Figura 4 también muestra que la solución computada por *qpsolvers* arroja un número mayor de vectores de soporte (34 frente a 24 de *quadprog*). Esto podría indicar que la solución entregada por *qpsolvers* puede estar capturando mayores detalles de los datos y puede no estar generalizando adecuadamente, esto podría explicar los resultados documentados en la Tabla 4 para la base de datos clasificación 2, donde se aprecia un mejor rendimiento utilizando *quadprog*. La diferencia observada en los resultados admite también una interpretación matemática, la SVM, aunque convexa, no es estrictamente convexa (no se puede garantizar que $\boldsymbol{P}$ sea definida positiva).

En consecuencia, el QP puede admitir múltiples soluciones equivalentes, es decir, diferentes vectores de soporte que llevan al mismo valor óptimo al problema expuesto en la ecuación 18, este hecho es validado en la Figura 4, donde por diferentes herramientas de cómputo se obtiene soluciones distintas que conducen al mismo mínimo global del QP. En síntesis, los resultados representados en la Figura 4, no expresan que el QP sea no convexo, sino que su solución no es única, lo cual es un resultado esperado al trabajar con matrices $\boldsymbol{P}$ semidefinidas positivas.

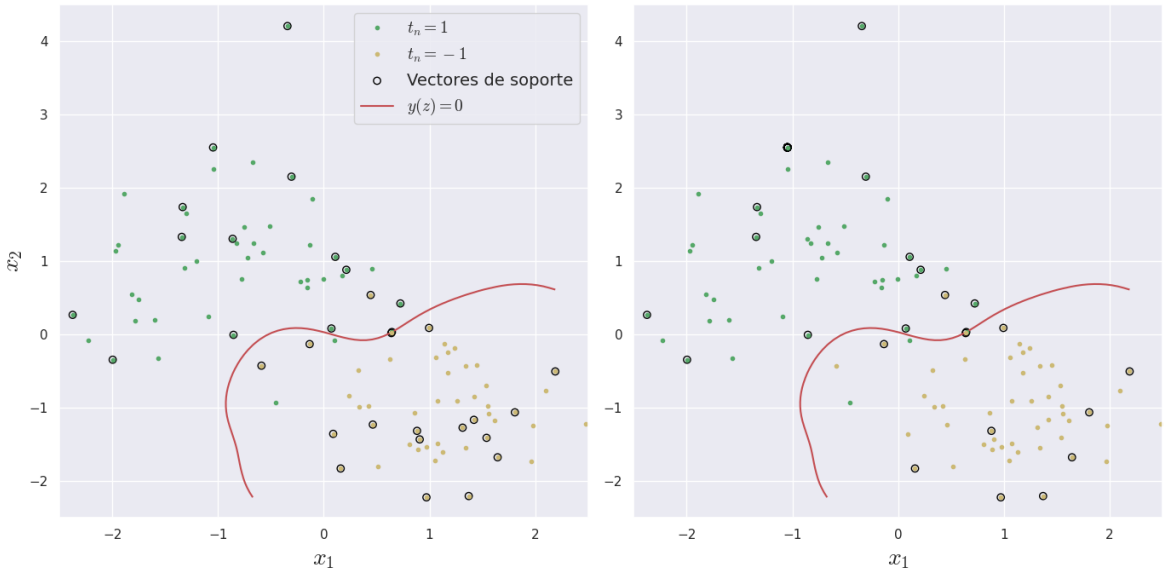

Figura 4: Curva de decisión y vectores de soporte caso clasificación binaria para la base de datos 2 clasificación. Gráfico izquierdo solución encontrada por *qpsolvers*¸ gráfico derecho solución *quadprog*.

Los resultados para la base de datos clasificación 1 se presentan en la Figura 5. Cabe destacar que la SVM logra separar adecuadamente las etiquetas mediante una curva de decisión no lineal. Este resultado, enseñado en la Figura 5, está en sintonía con los datos presentados en la Tabla 4, donde las métricas de desempeño sugieren que la SVM logra altos niveles de desempeños en los datos de prueba. Es importante señalar que se obtuvieron resultados similares con *quadprog*.

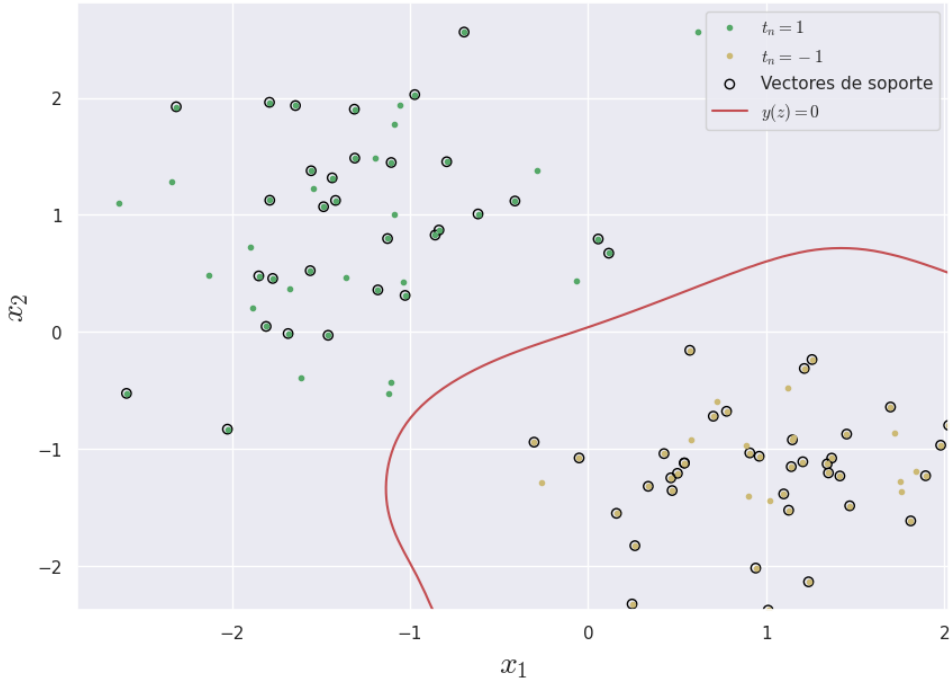

Figura 5: Solución obtenida con *qpsolvers* caso base datos clasificación 1.

En el caso de regresión, la Figura 6 presenta el comportamiento de los vectores de soporte para la base de datos de la Figura 2, se ilustra como la solución encontrada por *quadprog* arroja un número mayor de vectores de soporte (68 frente a 37 de *qpsolvers*), los vectores de soporte ilustrados en la Figura 6 deben interpretarse como los datos x_n que llevan a que $a_n - \hat{a}_n \neq 0$, es decir, los datos que tienen peso sobre la predicción dada por la ecuación 21. La diferencia entre el número de soportes en ambas soluciones puede indicar que la solución encontrada por *quadprog* abstrae más detalles de los datos, lo cual podría incluir el ruido presente en los datos y ser un síntoma de sobre ajuste.

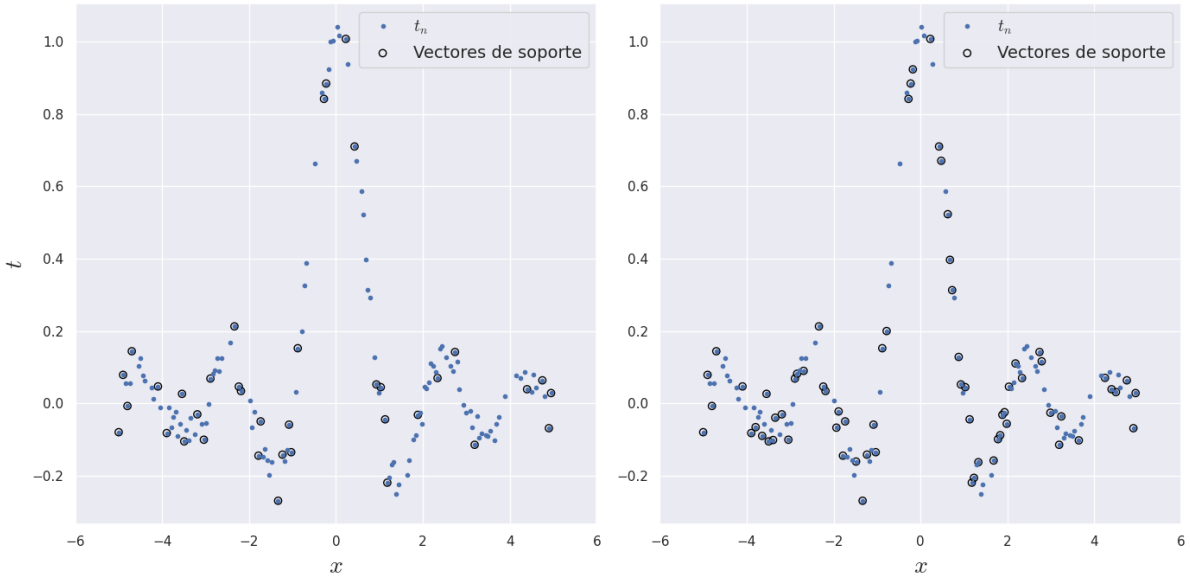

Figura 6: Vectores de soporte en regresión. Gráfico izquierdo solución encontrada por *qpsolvers*¸ gráfico derecho solución *quadprog*.

En la Figura 7, también se documenta el comportamiento de la predicción $y(\boldsymbol{z})$ versus los datos de prueba $\boldsymbol{y}$. En el gráfico de lado izquierdo, la predicción logra capturar el comportamiento de los datos de prueba $\boldsymbol{y}$, mientras que en el lado derecho se ilustra la dispersión de la predicción $y(\boldsymbol{z})$ frente a $\boldsymbol{y}$. Este resultado corrobora el valor reportado de $\rho_{y(\boldsymbol{z})\boldsymbol{y}}$ en la Tabla 4, dado que valores cercanos a 1 son indicadores de linealidad entre las variables analizadas, como se enseña en la Figura 7. Nótese además que la línea azul continua en la Figura 7 derecha es un indicador de qué tan cerca está la predicción del valor real. Por tanto, puntos negros sobre la línea azul indican una predicción satisfactoria, mientras que puntos muy alejados de la línea azul son un síntoma de una predicción pobre.

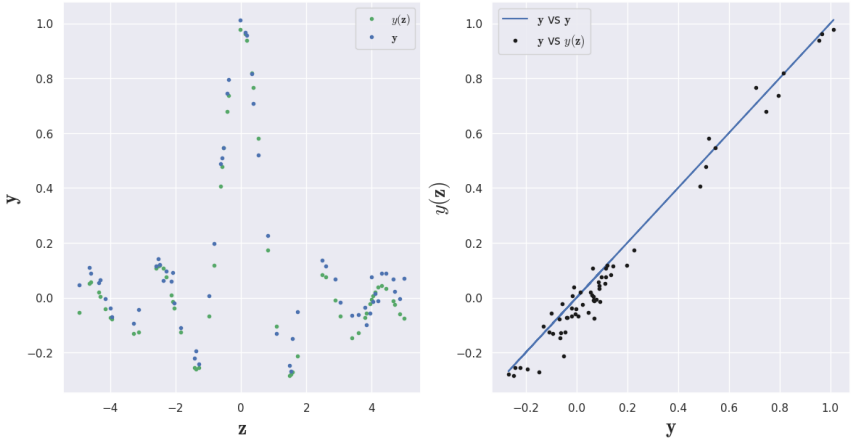

Figura 7: Comportamiento en la predicción caso regresión. Gráfico izquierdo comportamiento de la predicción frente a los datos de prueba. Gráfico derecho muestra la dispersión entre los datos de prueba y la predicción.

4. REPOSITORIO

Los datos y los códigos utilizados en este trabajo pueden ser consultados en: https://github.com/ChenaoB/Tutorial_SVM_Clasificacion_Regresion.

5. CONCLUSIONES

En este trabajo, se describió en detalle la implementación de una SVM en clasificación y regresión. A partir, de la expresión vectorial de la ecuación Lagrangiana dual, se estableció una relación entre este modelo y un QP. Del desarrollo matemático se identifican cuatro operadores básicos, en primer y segundo lugar, el producto interior y escalar entre dos vectores, en tercer lugar, el producto matricial, y por último el producto elemento a elemento o Hadamard. Estas operaciones permiten transformar la ecuación Lagrangiana dual en su forma vectorial equivalente. Como parte de la validación, se determinaron las expresiones Lagrangianas duales para una SVM en clasificación y regresión aplicando tales operadores. Estos resultados proporcionan una compresión más profunda de la relación que existe entre la SVM y un problema de programación cuadrática convexa.

En este trabajo también se ilustra cómo los operadores identificados permiten expresar la predicción $y(\boldsymbol{z})$ y la estimación del parámetro b en un formato vectorial equivalente. Esto es crucial, dado que facilita el uso de paquetes de software que optimizan el cálculo numérico al utilizar directamente vectores, como Matlab y la biblioteca *numpy* para Python.

En este trabajo se documentó un caso de estudio utilizando tres bases de datos sintéticas y dos paquetes de software para evaluar el desempeño de la SVM y el marco metodológico desarrollado. Basados en los resultados presentados en la Tabla 4 se muestra un alto rendimiento con ambas alternativas de computacionales. Además, las Figuras 4 y 5 proporcionan información detallada sobre aspectos fundamentales de la SVM. En el caso de clasificación binaria, se documenta el comportamiento del plano de decisión no lineal como una medida cualitativa para observar si la SVM logra separar adecuadamente las clases. Se destaca que este análisis sólo es posible únicamente para $N \leq 3$. Finalmente, los resultados de las Figuras 4, 5, 6 y 7 enseñan los vectores de soporte para clasificación y regresión, revelando una diferencia significativa en el número de vectores de soporte proporcionado por las herramientas computacionales, este aspecto es crucial para evaluar el desempeño de la SVM y debe ser analizado durante la fase de validación de los algoritmos.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] Cortes, C., y Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273-297. https://doi.org/10.1007/BF00994018

[2] Guido, R., Ferrisi, S., Lofaro, D., y Conforti, D. (2024). An Overview on the Advancements of Support Vector Machine Models in Healthcare Applications: A Review. *Information*, *15*(4), 235. https://doi.org/10.3390/info15040235

[3] Win, K., Sato, T., Tsuyuki, S. Application of Multi-Source Remote Sensing Data and Machine Learning for Surface Soil Moisture Mapping in Temperate Forests of Central Japan. *Information* **2024**, *15*, 485. https://doi.org/10.3390/info15080485

[4] Bishop, C. M., y Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer. https://doi.org/10.1117/1.2819119

[5] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), 121-167. https://doi.org/10.1023/A:1009715923555

[6] Suykens, J. A., y Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, *9*, 293-300. https://doi.org/10.1023/A:1018628609742

[7] Fung, G., y Mangasarian, O. L. (2001, August). Proximal support vector machine classifiers. in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 77-86). https://doi.org/10.1145/502512.502527

[8] Kumar, M. A., y Gopal, M. (2009). Least squares twin support vector machines for pattern classification. *Expert systems with applications*, *36*(4), 7535-7543. https://doi.org/10.1016/j.eswa.2008.09.066

[9] Kumar, M. A., Khemchandani, R., Gopal, M., y Chandra, S. (2010). Knowledge based least squares twin support vector machines. *Information Sciences*, *180*(23), 4606-4618. https://doi.org/10.1016/j.ins.2010.07.034

[10] Chen, J., y Ji, G. (2010, February). Weighted least squares twin support vector machines for pattern classification. In *2010 the 2nd international conference on computer and automation engineering (ICCAE)* (Vol. 2, pp. 242-246). IEEE. https://doi.org/10.1109/ICCAE.2010.5451483

[11] Chong, E. K., y Żak, S. H. (2013). *An introduction to optimization* (Vol. 75). John Wiley & Sons. https://doi.org/10.1002/9781118033340

[12] Saunders, C., Gammerman, A., y Vovk, V. (1998). Ridge Regression Learning Algorithm in Dual Variables. *International Conference on Machine Learning*. https://dl.acm.org/doi/10.5555/645527.657464

[13] Fung, G.M., Mangasarian, O.L. Multicategory Proximal Support Vector Machine Classifiers. *Mach Learn* 59, 77–97 (2005). https://doi.org/10.1007/s10994-005-0463-6

[14] Veropoulos, K., Campbell, C., y Cristianini, N. (1999, July). Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI* (Vol. 55, p. 60).

[15] Schölkopf, B., Smola, A. J., Williamson, R. C., y Bartlett, P. L. (2000). New support vector algorithms. *Neural computation*, *12*(5), 1207-1245. https://doi.org/10.1162/089976600300015565

[16] Khemchandani, R., y Chandra, S. (2007). Twin support vector machines for pattern classification. *IEEE Transactions on pattern analysis and machine intelligence*, *29*(5), 905-910. https://doi.org/10.1109/TPAMI.2007.1068

[17] Iranmehr, A., Masnadi-Shirazi, H., y Vasconcelos, N. (2019). Cost-sensitive support vector machines. *Neurocomputing*, *343*, 50-64. https://doi.org/10.1016/j.neucom.2018.11.099

[18] Pelckmans, K., Suykens, J. A., Van Gestel, T., De Brabanter, J., Lukas, L., Hamers, B., y Vandewalle, J. (2002). LS-SVMlab: a matlab/c toolbox for least squares support vector machines. *Tutorial. KULeuven-ESAT. Leuven, Belgium*, *142*(1-2). [Computer Software]. http://www.esat.kuleuven.be/sista/lssvmlab

[19] De Brabanter, K., Karsmakers, P., Ojeda, F., Alzate, C., De Brabanter, J., Pelckmans, K., y Suykens, J. A. (2010). *LS-SVMlab toolbox user's guide: version 1.7*. [Computer Software]. Katholieke Universiteit Leuven.

[20] Franc, V., y Hlavác, V. (2004). Statistical pattern recognition toolbox for Matlab. *Prague, Czech: Center for Machine Perception, Czech Technical University*. [Computer Software]. https://cmp.felk.cvut.cz/cmp/software/stprtool/

[21] Hsu, C. W. (2003). A Practical Guide to Support Vector Classification. *Department of Computer Science, National Taiwan University*. Disponible https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[22] Chang, C. C., y Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, *2*(3), 1-27. https://doi.org/10.1145/1961189.1961199

[23] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... y Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830. https://doi.org/10.48550/arXiv.1201.0490

[24] Lai, D., y Mani, N (2003). Technical Report MECSE-7-2003. Monash University, Departament of Electrical and Computer System Engineering.

[25] Horn, R. A., y Johnson, C. R. (2012). *Matrix analysis*. Cambridge university press. https://doi.org/10.1017/CBO9780511810817

[26] Boyd, S., y Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. https://doi.org/10.1017/CBO9780511804441

[27] Piccialli, V., y Sciandrone, M. (2022). Nonlinear optimization and support vector machines. *Annals of Operations Research*, *314*(1), 15-47. https://doi.org/10.1007/s10479-022-04655-x

[28] Du, K. L., Jiang, B., Lu, J., Hua, J., y Swamy, M. N. S. (2024). Exploring kernel machines and support vector machines: Principles, techniques, and future directions. *Mathematics*, *12*(24), 3935. https://doi.org/10.3390/math12243935

[29] Schölkopf, B., y Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press. https://doi.org/10.7551/mitpress/4175.001.0001

[30] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861-874. https://doi.org/10.1016/j.patrec.2005.10.010

[31] Goodfellow, I. (2016). *Deep learning* (Vol. 196). MIT press. http://www.deeplearningbook.org/

[32] Monteiro, N. R., Oliveira, J. L., y Arrais, J. P. (2022). DTITR: End-to-end drug–target binding affinity prediction with transformers. *Computers in Biology and Medicine*, *147*, 105772. https://doi.org/10.1016/j.compbiomed.2022.105772

[33] Caron, S., Arnström, D., Bonagiri, S., Dechaume, A., Flowers, N., Heins, A., Ishikawa, T., Kenefake, D., Mazzamuto, G., Meoli, D., O'Donoghue, B., Oppenheimer, A. A., Pandala, A., Quiroz Omaña, J. J., Rontsis, N., Shah, P., St-Jean, S., Vitucci, N., Wolfers, S., Yang, F., ... Khalil, A. (2024). *qpsolvers: Quadratic Programming Solvers in Python* (Version 4.3.3) [Computer software]. LGPL-3.0. https://github.com/qpsolvers/qpsolvers

[34] Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., y Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2

[35] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science & engineering*, *9*(03), 90-95. https://doi.org/10.1109/MCSE.2007.55

[36] Waskom, M. L. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021