UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

Introducción a las Máquinas de Vectores de Soporte para el Análisis de Sentimientos

Autor: Udane Viñas Templado

Directors: Dr. Nahuel Norberto Statuto i Dr. Oriol Pujol Vila
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 15 de enero de 2025

Abstract

This work explores Support Vector Machines (SVM), a key tool in machine learning. Through both theoretical and practical analysis, it examines their relationship with the Perceptron algorithm, optimization, and convexity—fundamental concepts for understanding how SVMs address complex classification problems. It delves into the mathematical principles underlying the model, such as primal and dual problems, Lagrangian functions, and duality theorems, which enable the optimization of both linear and nonlinear models.

The practical component of the work applies SVM to classify opinions in reviews of mobile phones, determining their polarity. This application illustrates how the model can process large volumes of data and be useful for tasks such as sentiment analysis on review platforms. The choice of SVM is not because it is the best approach for the analysis, but rather due to its ability to illustrate the theory in a simple way.

The aim of this work is to integrate the theory and practice of SVM, providing an understanding of its foundations and implementation, laying the groundwork for future developments in the classification of complex data, and demonstrating its potential in machine learning.

Resumen

Este trabajo explora las Máquinas de Soporte Vectorial (SVM), una herramienta clave en el aprendizaje automático. A través de un análisis teórico y práctico, se examina su relación con el algoritmo del Perceptrón, la optimización y la convexidad, conceptos fundamentales para entender cómo estas abordan problemas complejos de clasificación. Se profundiza en los principios matemáticos que sustentan el modelo, como los problemas primal y dual, las funciones Lagrangianas y los teoremas de dualidad, que permiten optimizar modelos tanto lineales como no lineales.

El componente práctico del trabajo aplica las SVM para clasificar opiniones de reseñas sobre teléfonos móviles, determinando su polaridad. Esta aplicación ilustra cómo el modelo puede procesar grandes volúmenes de datos y ser útil en tareas como el análisis de sentimientos en plataformas de reseñas. La elección de las SVM no responde a que sea el mejor enfoque para el análisis, sino por su capacidad para ilustrar la teoría de manera sencilla.

El objetivo del trabajo es integrar la teoría y la práctica de las SVM, proporcionando una comprensión de sus fundamentos y su implementación, sentando las bases para futuros desarrollos en la clasificación de datos complejos y demostrando el potencial de estas en el aprendizaje automático.

2020 Mathematics Subject Classification. 49K99, 52A41, 68T09, 62H30

Agradecimientos

Este trabajo, al igual que mi carrera, ha sido posible gracias al apoyo constante de mi familia y amigos. Agradezco la motivación y paciencia que me han brindado.

A mi **familia**, por enseñarme el valor del esfuerzo y la dedicación, y en especial a mi hermana Carla, por su apoyo incondicional. Gracias por estar siempre a mi lado y darme ánimos cuando los he necesitado.

A mis **amigos**, por acompañarme en esta aventura y por su apoyo durante todos estos años. Ha sido un placer compartir esta etapa con ellos, tanto académicamente como personalmente.

Y finalmente, a mis tutores, **Nahuel Norberto Statuto**, por su atención, orientación y disponibilidad desde el inicio de este trabajo. Y a **Oriol Pujol Vila**, por su consejo en la elección del tema, ya que fue crucial para darle dirección y sentido al proyecto.

A todos ellos, quería darles las gracias.

Índice general

1. Introducción 1
2. Teoría de la Optimización 3
   - 2.1. Introducción 3
   - 2.2. Conceptos Básicos 4
   - 2.3. Conjuntos y Funciones Convexas 7
   - 2.4. Optimización Convexa 10
     - 2.4.1. Teorema de KKT, condiciones y dualidad 11
3. Fundamentos de la SVM para la clasificación binaria 16
   - 3.1. Algorismo del Perceptrón 16
   - 3.2. Algorismo de las Máquinas de Soporte Vectorial 20
     - 3.2.1. Casos separables linealmente 21
     - 3.2.2. Casos cuasi-separables linealmente 26
     - 3.2.3. Casos no separables linealmente 29
4. Caso Práctico: Aplicación de las SVM 35
   - 4.1. Introducción 35
   - 4.2. Metodología 36
   - 4.3. Evaluación y resultados 42
5. Conclusiones 49

1. Introducción

Este trabajo tiene como **objetivo principal** explorar una de las herramientas más relevantes en el campo del aprendizaje automático: las Máquinas de Soporte Vectorial (SVM, por sus siglas en inglés). A lo largo de este proyecto, se profundiza en sus fundamentos teóricos, desvelando su vínculo con la optimización, un concepto clave para entender cómo estas técnicas abordan y resuelven problemas complejos de clasificación. Se examina, además, la aplicabilidad de las SVM en diversos escenarios, destacando su capacidad para manejar grandes volúmenes de datos y su versatilidad en situaciones no lineales. Estas máquinas han revolucionado la forma en la que se tratan diferentes problemas tanto lineales como no lineales, y a través de este análisis teórico, se ofrecen conocimientos básicos para comprender cómo funcionan y por qué son tan eficaces.

Lo fascinante es cómo combinan la teoría matemática con la resolución de problemas reales. Y aquí es donde cobra relevancia la optimización convexa, un concepto que no solo se usa para entender la base matemática de estas, sino también para resolver los problemas que surgen en aplicaciones prácticas. A medida que se exploran conceptos clave como los problemas primal y dual, a través de las funciones Lagrangianas y los teoremas de dualidad, no solo se descubre la teoría, sino cómo se puede aplicar a situaciones del mundo real.

El **componente práctico** de este trabajo se centra en aplicar los conocimientos adquiridos para clasificar opiniones extraídas de reseñas sobre teléfonos móviles. El desafío consiste en determinar si estas reseñas son positivas o negativas, empleando las Máquinas de Soporte Vectorial como herramienta para procesar y clasificar grandes volúmenes de datos de manera eficiente y precisa. Aunque, para este caso, el uso de este modelo no sea la opción más óptima, su objetivo es mostrar de forma clara y sencilla el funcionamiento de las SVM. En este sentido, el propósito de estudiar este ejemplo es ilustrar la teoría en acción y destacar la aplicabilidad del modelo en tareas de análisis de sentimientos.

La **estructura** de este proyecto está pensada para guiar al lector desde la teoría más fundamental a la más avanzada. En el **primer capítulo**, se aborda la *teoría de la optimización*, un tema crucial que sirve como marco para entender las SVM. Se inicia con una introducción general que muestra por qué la optimización es el corazón de estas y cómo se aplica al aprendizaje automático. A continuación, se exploran los conceptos básicos de la optimización, presentando teoremas fundamentales como el de Weierstrass, que garantiza la existencia de soluciones óptimas en problemas matemáticos y los conceptos de *conjuntos y funciones convexas*, explicando por qué estas estructuras matemáticas son tan poderosas cuando se trata de encontrar soluciones eficientes. La última sección de este capítulo trata sobre *optimización convexa*, donde se descubre cómo los problemas primal y dual se relacionan y cómo las funciones Lagrangianas y los teoremas de dualidad permiten resolver estos problemas complejos.

El **segundo capítulo** se centra en los *fundamentos de las SVM para la clasificación binaria*. En primer lugar, se explora el *algoritmo del Perceptrón*, una de las herramientas más antiguas y fundamentales en el campo de la clasificación. A partir de allí, se introducen las SVM, mostrando cómo superan sus limitaciones mediante su capacidad para manejar tres casos distintos: datos separables, datos casi separables y datos no separables. Esta sección es clave para entender la versatilidad del modelo y su capacidad para adaptarse a diferentes tipos de datos.

El **tercer capítulo**, *Caso Práctico*, pone en práctica lo aprendido al abordar un problema concreto: la clasificación de opiniones sobre teléfonos móviles. En primer lugar, se introduce el *objetivo* y se justifica la elección de este caso específico. Seguidamente, se inicia un proceso de *recolección de datos* a partir de diversas fuentes, lo que permite obtener las opiniones necesarias para el análisis. A continuación, se realiza el *preprocesamiento de los datos*, transformándolos en un formato adecuado que permita su uso en el modelo. Una vez los datos están listos, se pasa a la *implementación del modelo*, en la que se ajustan los parámetros para optimizar su rendimiento, y se continúa con la fase de *entrenamiento*, donde este se perfecciona mediante técnicas de optimización. Con el modelo ya entrenado, se procede a su *evaluación*, donde se analizan métricas clave como la precisión, la sensibilidad y la puntuación-F1 para determinar su efectividad. Finalmente, se concluye con *propuestas de mejora* para ajustarlo y obtener mejores resultados en futuras ampliaciones del trabajo.

Este proyecto culmina en las **conclusiones**, un espacio para reflexionar sobre lo que se ha logrado y lo que aún queda por descubrir. Se comparan los resultados obtenidos con los objetivos iniciales y se analizan las lecciones aprendidas durante el proceso.

En **resumen**, este trabajo de fin de grado no solo tiene como propósito demostrar cómo las SVM pueden abordar problemas de clasificación, sino también proporcionar una comprensión de los principios matemáticos que las fundamentan. El objetivo final es adquirir una base sólida tanto en la teoría como en la aplicación práctica de estas en contextos del mundo real. Así, al concluir este recorrido, se invita al lector a profundizar sobre el potencial del modelo y su relevancia en el ámbito del aprendizaje automático.

Lector, ¿te animas a embarcarte en este fascinante viaje hacia el corazón del aprendizaje automático?

2. Teoría de la Optimización

2.1. Introducción

Para poder entender los conocimientos más específicos y avanzados de las **Máquinas de Vectores de Soporte** es fundamental establecer primero los conceptos básicos que sustentan la **teoría de la optimización**. Por ello, se empieza definiendo el concepto más general.

Los problemas abordados por las SVM corresponden a problemas de optimización [2] con restricciones lineales. La **teoría de la optimización** es un conjunto de resultados y métodos analíticos enfocados en encontrar el mejor candidato entre una colección de alternativas sin tener que enumerar todas las posibles soluciones. En estos problemas, se identifican tres elementos fundamentales:

Las variables de decisión: Son las incógnitas del problema, es decir, las cantidades que queremos determinar para optimizar la función objetivo. Por ejemplo, $x_1, x_2, \ldots, x_n$ pueden representar decisiones como cantidades de recursos asignados, producción de diferentes bienes, o cualquier otro valor que influya en el objetivo del problema.

Las restricciones: Son las condiciones o limitaciones que deben cumplirse durante el proceso de optimización. Estas restricciones pueden ser ecuaciones o desigualdades que delimitan las posibles soluciones. Por ejemplo:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1, \\ &a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2, \\ &x_1, x_2, \ldots, x_n \geq 0, \end{aligned}$$

donde los coeficientes a_{ij} y los términos b_i representan los límites del problema.

La función objetivo $f(x_1, \ldots, x_n)$: Es la función que queremos maximizar o minimizar, dependiendo del tipo de problema.

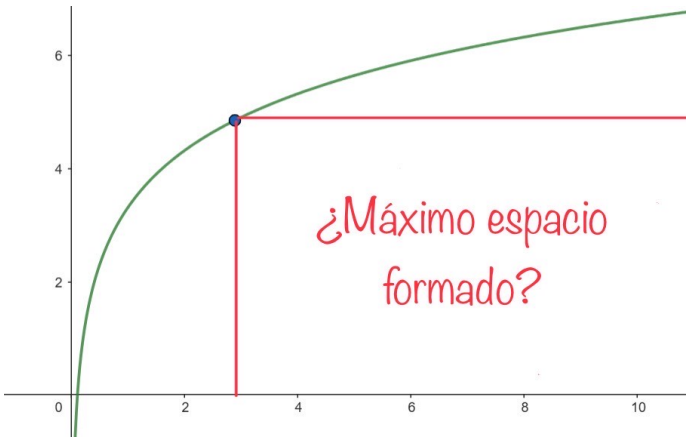

Figura 1: Ejemplo de problema de optimización típico [Fuente propia]

El **objetivo general** de la optimización es encontrar los valores de las variables de decisión que cumplan con las restricciones impuestas y, al mismo tiempo, **minimizen** o

[2] Un proceso de optimización consiste en buscar la mejor solución posible de entre todas las posibles soluciones que existen para resolver un problema.

maximizen la función objetivo. El planteamiento general de un problema de optimización es el siguiente:

$$\begin{aligned} \text{Optimizar} \quad & f(x_1, \ldots, x_n), \\ \text{sujeto a} \quad & \text{restricciones.} \end{aligned}$$

La siguiente sección introduce y analiza los elementos esenciales de la **teoría de la optimización**: las propiedades de los conjuntos, de las funciones y las herramientas teóricas que permiten su estudio. Estos conceptos iniciales proporcionan el marco necesario para profundizar en técnicas y resultados más concretos en capítulos posteriores.

2.2. Conceptos Básicos

El **Problema de Programación No Lineal** (PPNL) es una de las ramas más importantes de la optimización. Este problema surge cuando se busca maximizar o minimizar una función objetivo, que puede ser no lineal, sujeta a restricciones que involucran igualdades y desigualdades.

El objetivo principal del PPNL es encontrar soluciones que cumplan todas las restricciones y optimicen la función objetivo, garantizando que los resultados sean aplicables en contextos prácticos. Para lograrlo, se requiere una comprensión de los conceptos fundamentales, como soluciones factibles, conjuntos factibles y puntos extremos [3] , los cuales serán presentados en esta sección.

Definición 2.1 (**PPNL**). *Sea $(x_1, \ldots, x_n) \in \mathbb{R}^n$ el vector de variables de decisión y $A \subset \mathbb{R}^n$ el conjunto de partida donde se define el problema. El problema fundamental de optimización estática, conocido como* ***PPNL*** *(Problema de Programación No Lineal), se define como:*

$$\begin{aligned} \textit{Optimizar} \quad & f(x_1, \ldots, x_n), \quad (x_1, \ldots, x_n) \in A \subset \mathbb{R}^n, \\ \textit{sujeto a} \quad & h_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, m, \\ \textit{sujeto a} \quad & g_j(x_1, \ldots, x_n) \leq 0, \quad j = 1, \ldots, p. \end{aligned} \tag{2.2.1}$$

donde $f, h_i, g_j : A \to \mathbb{R}$.

Para abordar este problema, se introduce el concepto de **solución factible**, que es fundamental para definir el **conjunto factible** y las condiciones de optimización.

Definición 2.2 (**Solución y conjunto factible**). *Sea $A \subset \mathbb{R}^n$ el conjunto de partida donde se define el problema. Diremos que:*

1. $\overline{x} = (\overline{x_1}, \ldots, \overline{x_n}) \in A \subset \mathbb{R}^n$ es una ***solución factible*** *del problema (2.2.1) si y solo si cumple:*

$$\begin{cases} h_i(\overline{x_1}, \ldots, \overline{x_n}) = 0, & i = 1, \ldots, m, \\ g_j(\overline{x_1}, \ldots, \overline{x_n}) \leq 0, & j = 1, \ldots, p, \end{cases}$$

donde $h_i, g_j : A \to \mathbb{R}$.

[3] En cálculo, un extremo es el valor máximo o mínimo que una función puede alcanzar en un intervalo dado.

*2. El **conjunto factible** Ω del problema (2.2.1) es el conjunto de todas las soluciones factibles:*

$$\Omega = \{\overline{x} \in A \subset \mathbb{R}^n : \overline{x} \text{ es solución factible}\}.$$

Dentro del conjunto factible, se buscan puntos que representen los **extremos** de la función objetivo. Estos puntos pueden ser locales o globales y constituyen el foco de los teoremas fundamentales de la optimización, como el **Teorema de Weierstrass**, que será introducido más adelante.

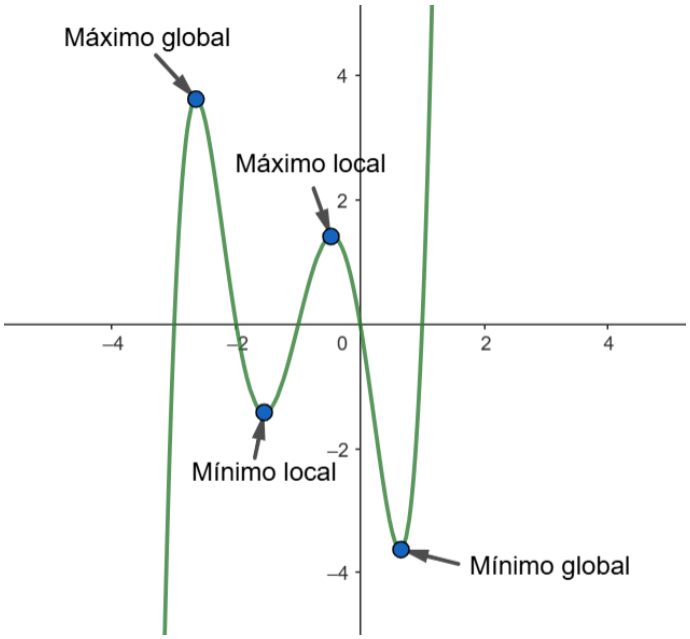

Figura 2: Extremos de una función [Fuente propia]

Definición 2.3 (**Mínimos y máximos locales y globales**). *Sea $\Omega \subset \mathbb{R}^n$ el conjunto de todas las soluciones factibles del problema de optimización. Diremos que:*

*1. $x^* = (x_1^*, \ldots, x_n^*) \in \Omega \subset \mathbb{R}^n$ es un **mínimo local** (o **máximo local**) de $f(x)$ en Ω si existe $\varepsilon > 0$ tal que, para cualquier $\overline{x} \in \Omega$ con $\overline{x} \neq x^*$, se cumple:*

$$\|\overline{x} - x^*\| < \varepsilon \quad \Rightarrow \quad \begin{cases} f(x^*) \leq f(\overline{x}) & \text{(mínimo local)}, \\ f(x^*) \geq f(\overline{x}) & \text{(máximo local)}. \end{cases}$$

*Si la desigualdad es estricta, es decir, $f(x^*) < f(\overline{x})$ para el mínimo local, o $f(x^*) > f(\overline{x})$ para el máximo local, entonces el mínimo o máximo es **estricto**.*

*2. $x^* = (x_1^*, \ldots, x_n^*) \in \Omega \subset \mathbb{R}^n$ es un **mínimo global** (o **máximo global**) de $f(x)$ en Ω si, para cualquier $\overline{x} \in \Omega$, con $\overline{x} \neq x^*$, se cumple:*

$$\begin{cases} f(x^*) \leq f(\overline{x}) & \text{(mínimo global)}, \\ f(x^*) \geq f(\overline{x}) & \text{(máximo global)}. \end{cases}$$

*Si la desigualdad es estricta, es decir, $f(x^*) < f(\overline{x})$ para el mínimo global, o $f(x^*) > f(\overline{x})$ para el máximo global, entonces el mínimo o máximo es **estricto**.*

Una vez identificados estos puntos extremos, se puede hablar de **soluciones óptimas** y **valor óptimo** . Específicamente, la solución óptima es aquella que maximiza o minimiza la función objetivo dentro del conjunto factible y el valor óptimo es el valor de la función objetivo en la solución óptima.

Definición 2.4 (**Solución óptima y valor óptimo**). *Diremos que $x^* \in \Omega \subset \mathbb{R}^n$ es una* ***solución óptima*** *del problema (2.2.1) si cumple una de las siguientes condiciones:*

1. *Si el objetivo es minimizar, x^* es el mínimo global del problema (2.2.1).*
2. *Si el objetivo es maximizar, x^* es el máximo global del problema (2.2.1).*

En este caso, $f(x^*)$ será el ***valor óptimo*** *de la función objetivo.*

El siguiente teorema es relevante para asegurar la existencia de soluciones óptimas en ciertos problemas de optimización.

Teorema 2.5 (**Teorema de Weierstrass** [5]). *Sea $f(x)$ una función continua sobre un conjunto compacto $K \subset \mathbb{R}^n$. Entonces, para el problema de optimización:*

$$\begin{aligned} &\textit{Optimizar} \quad f(x), \\ &\textit{sujeto a} \quad x \in K, \end{aligned}$$

existen $x^*_{\text{mín}}, x^*_{\text{máx}} \in K$ tales que:

$$f(x^*_{\text{mín}}) = \min_{x \in K} f(x), \quad f(x^*_{\text{máx}}) = \max_{x \in K} f(x).$$

Demostración. Sabemos que $K \subset \mathbb{R}^n$ es compacto. Según el Teorema de Heine-Borel [10], un conjunto en $\mathbb{R}^n$ es compacto si y solo si es cerrado y acotado. Por lo tanto, K debe ser cerrado y acotado. Además, dado que $f(x)$ es continua, por el Teorema de continuidad en conjuntos compactos [11], $f(x)$ alcanza su valor máximo y mínimo en K. Es decir, existen puntos $x^*_{\text{mín}}, x^*_{\text{máx}} \in K$ tales que:

$$f(x^*_{\text{mín}}) = \min_{x \in K} f(x), \quad f(x^*_{\text{máx}}) = \max_{x \in K} f(x).$$

La continuidad de $f(x)$ garantiza que estos valores máximos y mínimos se logren en K. □

Este resultado teórico garantiza la existencia de valores óptimos para una función continua definida sobre un conjunto compacto. Este constituye una base para avanzar hacia el estudio de técnicas más especializadas y estructuradas en el campo de la optimización.

En los capítulos siguientes se profundiza en la **teoría de la optimización convexa**. Para ello, se centra la atención en conceptos clave como conjuntos convexos y funciones convexas, elementos fundamentales para comprender y resolver problemas dentro de este marco teórico.

La optimización convexa destaca no solo por su riqueza teórica, sino también por su relevancia práctica. Una de sus propiedades más notables es que cualquier mínimo local en un problema convexo es también un mínimo global, lo que simplifica considerablemente su análisis y resolución desde una perspectiva computacional.

2.3. Conjuntos y Funciones Convexas

Definición 2.6 (**Conjunto convexo**). *Se dice que un conjunto $C \subset \mathbb{R}^n$ es* ***convexo*** *si, para cualesquiera $x, y \in C$ y cualquier $\lambda \in [0,1]$, se cumple que:*

$$\lambda x + (1-\lambda)y \in C.$$

Definición 2.7 (**Función convexa**). *Dado un conjunto convexo $C \subset \mathbb{R}^n$, diremos que una función $f : \mathbb{R}^n \to \mathbb{R}$ es* ***convexa*** *sobre C si, para cualesquiera $x, y \in C$ y cualquier $\lambda \in [0,1]$, se cumple:*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$$

Si la desigualdad es estricta (es decir, $f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$ para $x \neq y$), entonces f se dice ***estrictamente convexa****.*

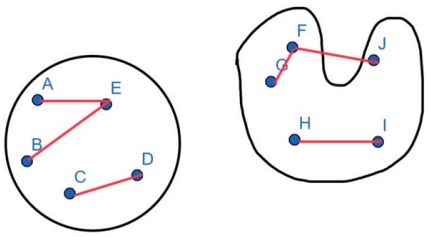

Figura 3: Ejemplos de conjunto convexo (primer conjunto) y conjunto no convexo (segundo conjunto) [Fuente propia]

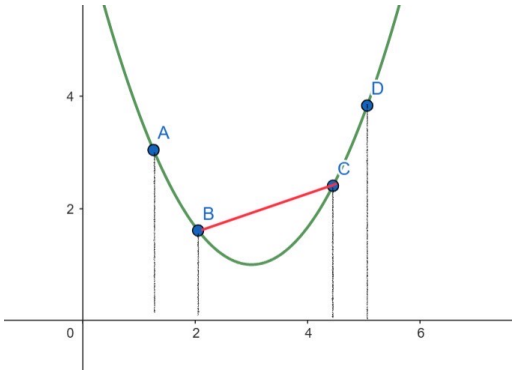

Figura 4: Ejemplo de función convexa [Fuente propia]

Es decir, un conjunto es convexo si cualquier combinación convexa de dos puntos dentro del conjunto también pertenece a él. Esta propiedad es clave en la definición de funciones convexas y en la formulación de problemas de optimización, ya que garantiza que el recorrido entre dos puntos dentro del conjunto no sale del conjunto.

Y una función convexa es aquella en la que el segmento [4] de línea que conecta dos puntos cualesquiera en su gráfica está siempre por encima o sobre la curva de la función.

En el gráfico anterior la **curva** representa una función convexa. Los puntos A, B, C, D están sobre la gráfica de la función, correspondiendo a diferentes valores dentro del dominio. El **segmento** BC conecta los puntos B y C. La convexidad garantiza que cualquier punto intermedio del segmento BC tiene una altura mayor o igual que el valor correspondiente de la curva.

Estas definiciones están directamente relacionadas con la teoría de optimización convexa, como se verá en los siguientes teoremas, donde interesa asegurar que los mínimos de funciones convexas sobre conjuntos convexos son globales.

[4] Un segmento de línea es una porción de una línea recta que está delimitada por dos puntos específicos, denominados extremos del segmento.

Teorema 2.8 (**Teorema sobre el mínimo de funciones convexas**). *Sea $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ con Ω un conjunto convexo. Si $f(x)$ es convexa (o cóncava), entonces el conjunto T, donde $f(x)$ alcanza su máximo (mínimo), es convexo y cualquier mínimo (máximo) local de $f(x)$ es mínimo (máximo) global.*

Demostración. Definimos el conjunto T como el conjunto de puntos donde $f(x)$ alcanza su mínimo:

$$T = \left\{ x^* \in \Omega \mid f(x^*) = \min_{x \in \Omega} f(x) \right\}.$$

Luego, discutimos tres posibles casos:

1. **Si T es el conjunto vacío (sin mínimos relativos):** En este caso $f(x)$ no tiene mínimos relativos, entonces el teorema se cumple por la convexidad del conjunto vacío.

2. **Si T no es vacío:** Supongamos que $T \neq \emptyset$, es decir, existe al menos un punto $x^* \in T$. Si x^* es el único elemento de T, entonces T es un conjunto convexo porque cualquier combinación convexa de x^* con x^* sigue siendo x^*, es decir,

$$\lambda x^* + (1-\lambda)x^* = x^*, \quad \text{para cualquier } \lambda \in [0,1].$$

3. **Si existen al menos dos elementos $x^*, y^* \in T$:** Sabemos que para cualquier $\lambda \in [0,1]$, usando la convexidad de $f(x)$, se cumple que

$$f(\lambda x^* + (1-\lambda)y^*) \leq \lambda f(x^*) + (1-\lambda) f(y^*) = \lambda \min_{x \in \Omega} f(x) + (1-\lambda) \min_{x \in \Omega} f(x) = \min_{x \in \Omega} f(x).$$

   Por lo tanto, $f(\lambda x^* + (1-\lambda)y^*) = \min_{x \in \Omega} f(x)$, lo que demuestra que $\lambda x^* + (1-\lambda)y^* \in T$, es decir, T es convexo.

De esta manera, se concluye que el conjunto T, donde se alcanza el mínimo de $f(x)$, es convexo. Además, cualquier mínimo local de $f(x)$ es también un mínimo global. □

A continuación, se presentan dos lemas adicionales que completan esta teoría, garantizando que los mínimos globales de una función convexa son únicos.

Lema 2.9. *Si x^*, y^* son dos puntos de mínimo [máximo] global de una función convexa [cóncava] $f(x)$, con $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ y Ω convexo, entonces para todo $\lambda \in [0,1]$, los puntos definidos como*

$$z^*(\lambda) = \lambda x^* + (1-\lambda)y^*$$

también son mínimos [máximos] globales de $f(x)$.

Demostración. Supongamos que f es convexa. Entonces, para cualquier $\lambda \in [0,1]$, se tiene que:

$$f(z^*(\lambda)) = f(\lambda x^* + (1-\lambda)y^*) \leq \lambda f(x^*) + (1-\lambda) f(y^*).$$

Como x^* y y^* son mínimos globales, $f(x^*) = f(y^*)$, y por lo tanto:

$$f(z^*(\lambda)) \leq f(x^*).$$

Dado que x^* es el mínimo global, esto implica que $z^*(\lambda) \in \Omega$, y por lo tanto, $f(z^*(\lambda)) = f(x^*)$. Así, los puntos de la forma $z^*(\lambda)$ son todos mínimos globales de $f(x)$. □

Lema 2.10. *Sea $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$, con Ω un conjunto convexo no vacío y $f(x)$ convexa [cóncava]. Sea x^* un mínimo [máximo] local de f. Entonces, si x^* es un mínimo [máximo] local estricto de $f(x)$ o si $f(x)$ es estrictamente convexa [cóncava] sobre Ω, se deduce que x^* es el único mínimo [máximo] global de $f(x)$ en Ω.*

Demostración. Si x^* es un mínimo local, entonces por el teorema (2.11) es un mínimo global. Si suponemos ahora que existe otro punto z^* mínimo de $f(x)$, por el lema (2.12) el mínimo también se alcanza en todo el segmento que une ambos puntos x^* y z^*, y esto contradice el hecho de que x^* sea estricto.

Supongamos ahora que x^* es un mínimo local de $f(x)$ y que $f(x)$ es estrictamente convexa. Como $f(x)$ es convexa, ya que es estrictamente convexa, x^* es un mínimo global. Si ahora suponemos que existe otro mínimo global z^*, por la convexidad estricta de $f(x)$ obtenemos:

$$f\left(\frac{1}{2}x^* + \frac{1}{2}z^*\right) < \frac{1}{2}f(x^*) + \frac{1}{2}f(z^*) = f(x^*),$$

y por tanto x^* no sería mínimo global. □

Por último, presentamos el teorema de existencia y unicidad de la solución, que establece que bajo ciertas condiciones, los problemas de optimización tienen solución y esta es única.

Teorema 2.11 (**Teorema de Existencia y Unicidad de Solución**)**.** *Supongamos que el problema de optimización es el siguiente:*

$$\begin{aligned} &\textit{Optimizar} \quad f(x_1, \ldots, x_n), \\ &\textit{sujeto a:} \\ &\begin{cases} h_i(x_1, \ldots, x_n) = 0, & i = 1, \ldots, m, \\ g_j(x_1, \ldots, x_n) \leq 0, & j = 1, \ldots, p, \\ (x_1, \ldots, x_n) \in A \subset \mathbb{R}^n, & \textit{(dominio de las variables).} \end{cases} \end{aligned}$$

Donde $f : \mathbb{R}^n \to \mathbb{R}$ es la función objetivo, $h_i(x_1, \ldots, x_n) = 0$ son las restricciones de igualdad, $g_j(x_1, \ldots, x_n) \leq 0$ son las restricciones de desigualdad, y el conjunto $A \subset \mathbb{R}^n$ es el dominio de las variables.

Si la función $f(x)$ es continua y convexa, las restricciones $h_i(x) = 0$ son funciones suaves y las restricciones $g_j(x) \leq 0$ son funciones convexas, y el conjunto de soluciones factibles es no vacío y compacto, entonces el problema de optimización tiene una solución óptima única.

Demostración. Este teorema establece las condiciones necesarias para que exista una solución única en problemas de optimización bajo restricciones. Primero, dado que las funciones de objetivo y las restricciones son continuas y convexas, y el conjunto de soluciones

factibles es compacto (por ejemplo, un conjunto cerrado y acotado), podemos aplicar el teorema de existencia de soluciones de Weierstrass. Este teorema asegura que existe al menos un punto en el conjunto factible que optimiza la función objetivo.

Ahora bien, si la función $f(x)$ es estrictamente convexa, la convexidad estricta garantiza que la función no tendrá más de un mínimo o máximo global. En este caso, el punto óptimo es único, ya que no puede haber más de un punto que logre el valor máximo o mínimo. □

Habiendo establecido los conceptos fundamentales de conjuntos y funciones convexas, es posible adentrarse en el **estudio de la optimización convexa**. Esta rama de la optimización se caracteriza por aprovechar las propiedades estructurales de la convexidad para garantizar soluciones globales y simplificar el análisis de los problemas.

En esta sección, se explora los principios que rigen los problemas de optimización convexa, así como su formulación y resolución. En particular, se introducen los conceptos de problema primal y problema dual, elementos centrales en esta teoría.

El análisis de los problemas primal y dual, junto con las condiciones de optimalidad que los vinculan, constituye una herramienta clave para resolver problemas complejos en diversos contextos prácticos. Esta sección sienta las bases para comprender y aplicar estos conceptos en escenarios más avanzados.

2.4. Optimización Convexa

Como se ha mencionado anteriormente, consideremos el **problema primal** en optimización convexa como:

$$\begin{aligned} \text{Minimizar} \quad & f(x), \\ \text{sujeto a} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, n. \end{aligned}$$

donde $f(x)$ es la función objetivo a minimizar, y las funciones $g_i(x)$ son las restricciones del problema. En este caso, la **solución primal** x^* será aquella que minimice $f(x)$ bajo las restricciones $g_i(x) \leq 0$ para $i = 1, \ldots, n$.

En el proceso de transformación a un problema dual, la **función de Lagrange** desempeña un papel fundamental. Esta nos permite incorporar las restricciones en una única función objetivo mediante el uso de los **multiplicadores de Lagrange**.

Definición 2.12 (**Función de Lagrange**)**.** *Dada una función objetivo $f(x)$ y un conjunto de restricciones $g_i(x) = 0$ para $i = 1, \ldots, m$, la **función de Lagrange**[8] se define como:*

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x),$$

*donde λ_i son los **multiplicadores de Lagrange**, que actúan como parámetros que penalizan el incumplimiento de las restricciones. Esta se utiliza para resolver problemas de optimización con restricciones.*

Una vez formulado el problema en términos del Lagrangiano, el siguiente paso consiste en maximizar respecto a los multiplicadores λ_i, mientras se minimiza respecto a x. Este proceso da lugar al **problema dual**.

De esta forma definimos el **problema dual** de la siguiente manera:

$$\begin{aligned} &\text{Maximizar} \quad \varphi(\lambda) = \inf_x L(x, \lambda), \\ &\text{sujeto a} \quad \lambda_i \geq 0, \quad i = 1, \ldots, n. \end{aligned}$$

Donde $\varphi(\lambda)$ es la función objetivo del problema dual, y la maximización de esta función sobre los multiplicadores λ_i nos da información sobre las soluciones óptimas del problema primal. La función $\inf_x L(x, \lambda)$ es el valor mínimo de la función Lagrangiana respecto a x, lo cual corresponde a resolver el problema primal para valores dados de λ_i

2.4.1. Teorema de KKT, condiciones y dualidad

El siguiente teorema establece la relación entre el problema primal (de minimización) y su problema dual (de maximización) mediante el uso del Lagrangiano.

Teorema 2.13 (**Teorema de Dualidad Lagrangiana**)**.** *Dado el problema primal de optimización:*

$$\begin{aligned} \textit{Minimizar} \quad & f(x), \\ \textit{sujeto a} \quad & g_i(x) \leq 0,\ i = 1, \ldots, m, \\ & h_j(x) = 0,\ j = 1, \ldots, p, \end{aligned}$$

donde $f(x)$ es la función objetivo, $g_i(x)$ son las restricciones de desigualdad, y $h_j(x)$ son las restricciones de igualdad.

*El **problema dual** asociado es el siguiente:*

$$\textit{maximizar}_{\lambda \geq 0, \mu} \quad \inf_x \mathcal{L}(x, \lambda, \mu) = \textit{maximizar}_{\lambda \geq 0, \mu} \quad \left(f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x) \right),$$

*donde $\lambda_i \geq 0$ y μ_j son los **multiplicadores de Lagrange** asociados a las restricciones de desigualdad y de igualdad, respectivamente.*

Las condiciones de Karush-Kuhn-Tucker (KKT) son fundamentales para establecer la equivalencia entre las soluciones del problema primal y su correspondiente problema dual.

Teorema 2.14 (**Teorema de Karush-Kuhn-Tucker (KKT)** [9])**.** *Consideramos el problema de optimización no lineal:*

$$\begin{aligned} \textit{Minimizar} \quad & f(x), \\ \textit{sujeto a} \quad & g_i(x) \leq 0,\ i = 1, \ldots, m, \end{aligned}$$

donde $f(x)$ y $g_i(x)$ son funciones diferenciables. Si x^* es un óptimo local y se cumplen ciertas condiciones regulares (como la cualificación de restricciones), entonces existen multiplicadores $\lambda_i \geq 0$ tales que se satisfacen las siguientes condiciones:

Condición de estacionaridad:

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) = 0.$$

Esta condición obliga a que el gradiente

Condición primal:

$$\lambda_i \nabla g_i(x^*) = 0, \quad \forall i = 1, \ldots, m$$

Esta condición obliga a que los multiplicadores de Karush-Kuhn-Tucker asociados a restricciones NO saturadas en x^* sean nulos. Veámoslo:

- *Si la restricción g_i está saturada en x^* ($g_i(x^*) = 0$), entonces:*

$$\lambda_i g_i(x^*) = 0 \quad \textit{para cualquier valor de} \quad \lambda_i.$$

- *Si la restricción g_i NO está saturada en x^* ($g_i(x^*) < 0$), entonces la única posibilidad para que $\lambda_i g_i(x^*) = 0$ es que:*

$$\lambda_i = 0.$$

Condición dual:

$$\lambda_i \geq 0, \quad \forall i = 1, \ldots, m \quad \textit{(si se trata de máximo)},$$

$$\lambda_i \leq 0, \quad \forall i = 1, \ldots, m \quad \textit{(si se trata de mínimo)}.$$

Cuando el punto satisface KKT para un máximo, todos los multiplicadores deben ser no negativos ($\lambda_i \geq 0$). Cuando el punto satisface KKT para un mínimo, todos los multiplicadores deben ser no positivos ($\lambda_i \leq 0$).

Condición de complementariedad:

$$g_i(x^*) \leq 0, \quad \forall i = 1, \ldots, m.$$

El punto x^* debe satisfacer todas las restricciones del problema, es decir, debe pertenecer al conjunto de soluciones factibles del problema.

El siguiente teorema, el teorema de dualidad, proporciona la relación entre el problema primal y su dual, garantizando que, bajo ciertas condiciones, ambas soluciones coinciden. Es por este motivo que es una herramienta clave en la resolución de problemas de optimización.

Teorema 2.15 (**Teorema de Dualidad en Optimización**).

Dualidad débil: *Para cualquier solución factible x^* del problema primal y cualquier (λ^*, μ^*) factibles en el problema dual, se cumple:*

$$f(x^*) \geq \mathcal{L}(x^*, \lambda^*, \mu^*),$$

donde $f(x^*)$ es el valor de la función objetivo en la solución óptima del problema primal, $\mathcal{L}(x^*, \lambda^*, \mu^*)$ es la función Lagrangiana evaluada en la solución primal x^* y en los multiplicadores λ^* y μ^*, que son los multiplicadores de Lagrange óptimos del problema dual.

Dualidad fuerte: *En problemas convexos donde se cumple la cualificación de restricciones (como las condiciones de Karush-Kuhn-Tucker), la solución óptima del problema primal coincide con la del problema dual:*

$$f(x^*) = \mathcal{L}(x^*, \lambda^*, \mu^*),$$

lo que significa que el valor de la función objetivo en la solución óptima del problema primal es igual al valor de la función Lagrangiana evaluada en los multiplicadores óptimos del problema dual.

Demostración. Para demostrar la dualidad débil, comenzamos con la formulación del problema primal y su correspondiente problema dual. Recordemos que la función Lagrangiana asociada al problema primal es:

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x),$$

donde $\lambda_i \geq 0$ y μ_j son los multiplicadores de Lagrange asociados a las restricciones $g_i(x) \leq 0$ y $h_j(x) = 0$ respectivamente.

El problema dual se obtiene al minimizar la función Lagrangiana con respecto a x, lo que da lugar a la función dual $\mathcal{D}(\lambda, \mu)$:

$$\mathcal{D}(\lambda, \mu) = \inf_x \mathcal{L}(x, \lambda, \mu).$$

El valor dual depende de los multiplicadores λ y μ, y el problema dual consiste en maximizar esta función dual con respecto a λ y μ, bajo la restricción de que $\lambda_i \geq 0$.

La dualidad débil establece que el valor óptimo del problema primal nunca es menor que el valor óptimo del problema dual. Es decir:

$$f(x^*) \geq \mathcal{D}(\lambda^*, \mu^*),$$

donde x^* es la solución óptima del problema primal y (λ^*, μ^*) son los multiplicadores óptimos del problema dual.

La dualidad fuerte se da cuando las condiciones de Karush-Kuhn-Tucker (KKT) se cumplen. En este caso, los óptimos del problema primal y el problema dual coinciden:

$$f(x^*) = \mathcal{D}(\lambda^*, \mu^*),$$

lo que implica que las soluciones primal y dual son equivalentes.

□

Cuando el número de variables en un problema de optimización es mayor que el número de restricciones, resulta más conveniente resolver primero el problema dual. A partir de su solución óptima, utilizando los multiplicadores asociados, se puede deducir la solución óptima del problema primal.

Para ilustrar esta relación entre los problemas primal y dual, se muestra el siguiente ejemplo [7]:

En este ejercicio, el problema primal P es un problema de maximización con tres variables de decisión (x, y, z) y dos restricciones lineales. Su correspondiente problema dual D es un problema de minimización con dos variables duales (u, v), cuyas restricciones están directamente relacionadas con las variables del problema primal.

$$
\begin{aligned}
&(P) \quad \text{Problema Primal} \\
\max \quad & f(x, y, z) = 4x + 2y + z \\
& 2x + y \leq 30, \\
& 2x + 2y + z \leq 40, \\
& x, y, z \geq 0.
\end{aligned}
\qquad
\begin{aligned}
&(D) \quad \text{Problema Dual} \\
\min \quad & \tilde{f}(u, v) = 30u + 40v \\
& 2u + 2v \geq 4, \\
& u + 2v \geq 2, \\
& v \geq 1, \\
& u, v \geq 0.
\end{aligned}
$$

La formulación del problema dual se obtiene aplicando la teoría de la dualidad. A continuación, se describen los pasos clave para construir el problema dual D a partir del problema primal P:

El objetivo del problema primal es maximizar $f(x, y, z) = 4x + 2y + z$, que depende de las variables x, y y z. Los coeficientes de la función objetivo corresponden a los términos de las restricciones en el dual.

Las restricciones en el problema primal son de tipo desigualdad, lo que nos lleva a introducir variables de Lagrange (o variables duales) en el problema dual.
- Para la restricción $2x + y \leq 30$, introducimos una variable dual $u \geq 0$.
- Para la restricción $2x + 2y + z \leq 40$, introducimos una variable dual $v \geq 0$.

La función objetivo del problema dual es una minimización, y se construye a partir de las constantes de las restricciones del primal:

$$\min \quad \tilde{f}(u, v) = 30u + 40v$$

Esto surge del hecho de que el valor de las restricciones del primal (30 y 40) son multiplicados por las respectivas variables duales u y v.

Las restricciones del problema dual se derivan de los coeficientes de las variables x, y y z en las restricciones del problema primal.

La primera restricción del primal es $2x + y \leq 30$. Para cada una de las variables x, y y z del primal, debemos escribir una desigualdad que involucre las variables duales u y v.

- Para x: Al sumar las contribuciones de x en ambas restricciones del primal ($2x$ en la primera y $2x$ en la segunda), obtenemos:

$2u + 2v \geq 4$ (debido al coeficiente de x en la función objetivo primal, que es 4).

- Para y: Al hacer lo mismo para y en las restricciones del primal (y en la primera y $2y$ en la segunda), obtenemos:

$u + 2v \geq 2$ (debido al coeficiente de y en la función objetivo primal, que es 2).

- Para z: Como z solo aparece en la segunda restricción, obtenemos la condición:

$v \geq 1$ (debido al coeficiente de z en la función objetivo primal, que es 1).

En el problema dual, las variables duales u y v deben ser no negativas, ya que las restricciones del primal son de tipo desigualdad "menor o igual".

Como podemos observar, es más conveniente resolver el problema dual, ya que involucra únicamente dos variables. Esta simplificación es precisamente la razón por la cual aplicamos esta transformación.

Puesto que:

$$\tilde{f}(0,2) = 80, \quad \tilde{f}(1,1) = 70,$$

el posible punto óptimo será el vértice $(1,1)$.

Se ve que el dominio no es compacto y que, por tanto, no puede aplicarse el *teorema de Weierstrass*. Sin embargo, es posible aplicar las condiciones de KKT al punto $(1,1)$. Los multiplicadores obtenidos son los siguientes:

$$\lambda_1 = 15, \quad \lambda_2 = 0 \quad \text{y} \quad \lambda_3 = 10.$$

Así pues, se verifica que el punto $(1,1)$ es mínimo global de (D). Además, podemos deducir que el punto óptimo del problema primal es $(15, 0, 10)$ y que el valor máximo es 70, que coincide con el valor mínimo del problema dual.

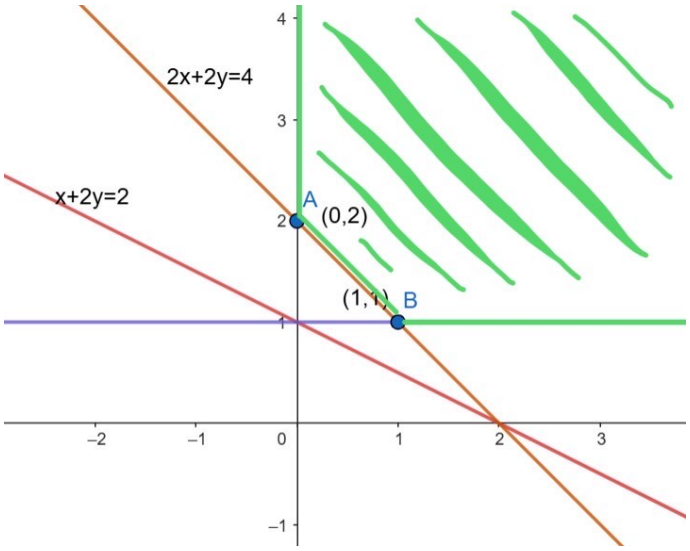

Figura 5: Imagen visual del problema [Fuente propia]

Los fundamentos teóricos establecidos permiten adentrarse en el estudio de las **Máquinas de Vectores de Soporte**, una de las técnicas más poderosas y ampliamente utilizadas para la clasificación de datos. Estas máquinas buscan determinar un hiperplano que separe de manera óptima dos clases, maximizando el margen entre ellas.

En este contexto, la optimización convexa desempeña un papel crucial, ya que proporciona el marco matemático necesario para formular y resolver el problema de maximización del margen. Al mismo tiempo, permite incorporar restricciones que minimicen los errores de clasificación en caso de que los datos no sean completamente separables.

El siguiente capítulo se centra en cómo la teoría de la optimización convexa respalda el diseño y la resolución eficiente de las SVM, estableciendo una conexión directa entre los conceptos teóricos y su aplicación práctica.

3. Fundamentos de la SVM para la clasificación binaria

3.1. Algorismo del Perceptrón

Introducción al algoritmo

El **Perceptrón** es un algoritmo de aprendizaje supervisado que se utiliza principalmente para problemas de clasificación binaria. Su objetivo principal es encontrar un **hiperplano de separación** que permita clasificar datos en dos clases distintas. Este algoritmo es considerado uno de los primeros enfoques en el campo del aprendizaje automático y se utiliza como modelo base para otros métodos de clasificación lineales, como la **Máquina de Vectores de Soporte**.

El modelo fue propuesto por el científico **Frank Rosenblatt** en 1957. Originalmente, el algoritmo fue diseñado para emular cómo los seres humanos aprenden a clasificar objetos visuales, y su desarrollo marcó un hito importante en la historia de la **inteligencia artificial** (IA) [5]. En sus primeras implementaciones, el Perceptrón estaba basado en una red neuronal simple, compuesta por una sola capa de nodos (neuronas), lo que le permitió realizar tareas de clasificación lineal.

A pesar de sus limitaciones, que serán comentadas posteriormente, el Perceptrón tuvo un impacto significativo al dar origen a investigaciones más profundas sobre redes neuronales [6] y aprendizaje automático. Su desarrollo también fue clave para la posterior creación de métodos más avanzados, como las redes neuronales multicapa [7] y los algoritmos de optimización más sofisticados utilizados en la actualidad.

Objetivo del Perceptrón

El Perceptrón busca encontrar un hiperplano definido por los **pesos** w y un **sesgo** b que divida correctamente dos clases de datos $y_i \in \{-1, +1\}$. Este hiperplano se describe como:

$$\langle w, x \rangle + b = 0,$$

donde w y b se ajustan iterativamente para minimizar los errores de clasificación.

Explicación del algoritmo del Perceptrón

El **algoritmo** ajusta los pesos y el sesgo basándose en los errores de clasificación, siguiendo las siguientes reglas:

[5] La inteligencia artificial (IA) es una rama de la informática que se centra en la creación de sistemas y máquinas capaces de realizar tareas que, normalmente, requieren de la inteligencia humana.

[6] Las redes neuronales son un modelo de computación inspirado en el cerebro humano que se utiliza principalmente en el campo del aprendizaje automático.

[7] Las redes neuronales multicapa (MLP, por sus siglas en inglés, Multilayer Perceptron) son un tipo de red neuronal artificial que consta de múltiples capas de neuronas: una capa de entrada, una o más capas ocultas y una capa de salida.

Los pesos w y el sesgo b se actualizan únicamente cuando un punto de entrenamiento está mal clasificado. La **regla de actualización** es:

$$w \leftarrow w + \eta y_i x_i \quad \text{si} \quad (y_i w \cdot x_i + b) \leq 0,$$

$$b \leftarrow b + \eta y_i \quad \text{si} \quad (y_i w \cdot x_i + b) \leq 0,$$

donde:

- w: **Vector de pesos** que define la orientación del hiperplano.
- η: **Tasa de aprendizaje**, controla la magnitud de los ajustes.
- y_i: **Etiqueta del punto** x_i, $+1$ o -1.
- x_i: **Vector de características** del punto de entrenamiento.

El **flujo del algoritmo** es el siguiente:

1. Inicializar w y b con valores pequeños o aleatorios.
2. Para cada punto de entrenamiento (x_i, y_i):
   - Calcular $y_i\big(w \cdot x_i + b\big)$.
   - Si $y_i\big(w \cdot x_i + b\big) \leq 0$, actualizar w y b según las reglas correspondientes.
3. Repetir hasta que no haya errores de clasificación o se alcance un máximo número de iteraciones.

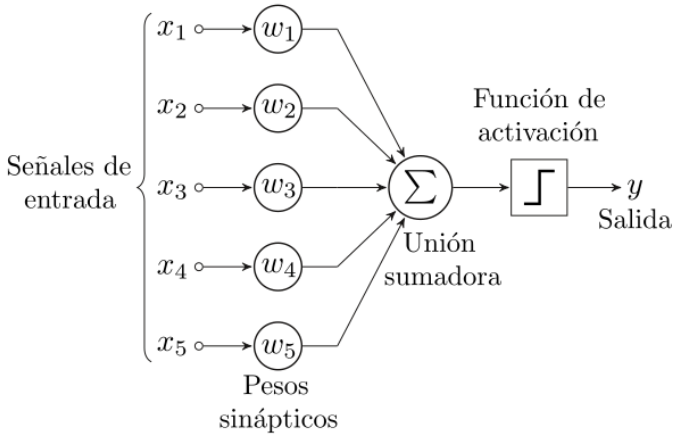

Figura 6: Diagrama de un perceptrón con cinco señales de entrada [18]

Para entender mejor el funcionamiento del algoritmo se proporciona el análisis de un caso concreto: se considera un conjunto de datos de entrenamiento con una sola característica x y las correspondientes etiquetas de clase y, donde $y = 1$ es la clase positiva y $y = -1$ es la clase negativa:

h!

x_i	y_i
1	1
2	1
3	-1
4	-1

Cuadro 1: Ejemplo conjunto de datos de entrenamiento

Se utiliza el algoritmo anterior para clasificar los puntos de entrenamiento, ajustando los pesos en función de si la predicción es correcta o incorrecta. Iniciamos el problema con los valores siguientes:

$$w = 0, \quad b = 0, \quad \eta = 0,1$$

Iteración 1 (Primer dato: $x_1 = 1, y_1 = 1$):

Calculamos $w \cdot x_1 + b$:

$$w \cdot x_1 + b = 0 \cdot 1 + 0 = 0$$

La predicción es incorrecta (porque $y_1 \cdot (w \cdot x_1 + b) = 1 \cdot 0 = 0 \leq 0$), por lo que se actualiza el peso y el sesgo.

Actualización de w y b:

$$w \leftarrow w + \eta \cdot y_1 \cdot x_1 = 0 + 0{,}1 \cdot 1 \cdot 1 = 0{,}1$$

$$b \leftarrow b + \eta \cdot y_1 = 0 + 0{,}1 \cdot 1 = 0{,}1$$

Iteración 2 (Segundo dato: $x_2 = 2, y_2 = 1$):

Calculamos $w \cdot x_2 + b$:

$$w \cdot x_2 + b = 0{,}1 \cdot 2 + 0{,}1 = 0{,}3$$

La predicción es correcta (porque $y_2 \cdot (w \cdot x_2 + b) = 1 \cdot 0{,}3 = 0{,}3 > 0$), por lo que no se actualiza el peso ni el sesgo.

Iteración 3 (Tercer dato: $x_3 = 3, y_3 = -1$):

Calculamos $w \cdot x_3 + b$:

$$w \cdot x_3 + b = 0{,}1 \cdot 3 + 0{,}1 = 0{,}4$$

La predicción es incorrecta (porque $y_3 \cdot (w \cdot x_3 + b) = -1 \cdot 0{,}4 = -0{,}4 \leq 0$), por lo que se actualiza el peso y el sesgo.

Actualización de w y b:

$$w \leftarrow w + \eta \cdot y_3 \cdot x_3 = 0{,}1 + 0{,}1 \cdot (-1) \cdot 3 = 0{,}1 - 0{,}3 = -0{,}2$$

$$b \leftarrow b + \eta \cdot y_3 = 0{,}1 + 0{,}1 \cdot (-1) = 0{,}1 - 0{,}1 = 0$$

Iteración 4 (Cuarto dato: $x_4 = 4, y_4 = -1$):

Calculamos $w \cdot x_4 + b$:

$$w \cdot x_4 + b = -0{,}2 \cdot 4 + 0 = -0{,}8$$

La predicción es correcta (porque $y_4 \cdot (w \cdot x_4 + b) = -1 \cdot (-0{,}8) = 0{,}8 > 0$), por lo que no se actualiza el peso ni el sesgo.

Este es un ejemplo de cómo el Perceptrón realiza una iteración, calcula las predicciones, y actualiza el peso de acuerdo con los errores cometidos. El proceso se repite durante varias iteraciones hasta que los pesos convergen o se alcanza el número máximo de iteraciones predeterminado [12].

Convergencia del Algoritmo del Perceptrón

El teorema siguiente nos proporciona una información clave, si los datos son **linealmente separables**, el algoritmo encuentra un hiperplano. Sin embargo, si los datos **no son separables**, el algoritmo no converge.

Teorema 3.1 (**de convergencia del Perceptrón** [13])**.** *Si el conjunto de patrones de entrenamiento*

$$\{(x_1, z_1), (x_2, z_2), \ldots, (x_p, z_p)\}$$

es linealmente separable, entonces el Perceptrón simple encuentra una solución en un número finito de iteraciones.

Además del resultado anterior, este modelo presenta ciertas limitaciones: En primer lugar, no garantiza convergencia si los datos no son linealmente separables. En segundo lugar, tiende a sobreajustarse a los datos de entrenamiento, lo que afecta su capacidad de generalización. Por último, el Perceptrón devuelve un hiperplano sin considerar el margen de separación, lo que lo hace menos robusto frente a nuevos datos.

Relación entre el Perceptrón y el las Máquinas de Soporte Vectorial

El **Perceptrón** y el **SVM** comparten una conexión importante en su enfoque para resolver problemas de clasificación. Ambos algoritmos buscan encontrar un **hiperplano de separación** que permita dividir los datos en dos clases.

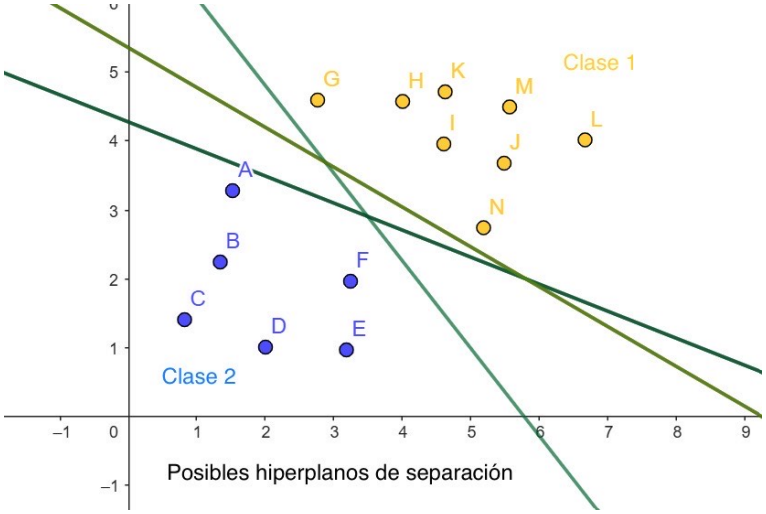

Figura 7: Hiperplanos de separación [Fuente propia]

Sin embargo, mientras que el Perceptrón se limita a buscar un hiperplano que clasifique correctamente los datos cuando son linealmente separables, las SVM van un paso más allá al identificar el hiperplano que **maximiza el margen** entre las clases. Esta característica convierte a las SVM en una extensión más avanzada y eficaz del Perceptrón, especialmente en tareas de clasificación complejas.

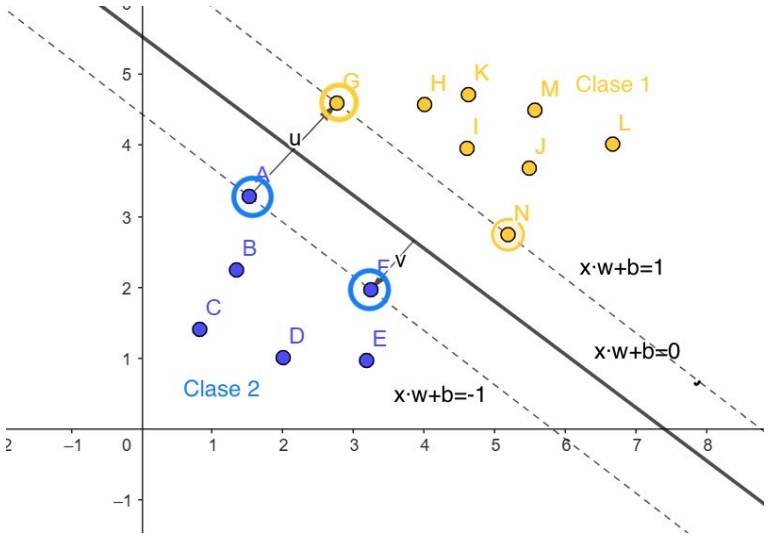

Figura 8: Hiperplano de separación óptimo [Fuente propia]

Para identificar el hiperplano óptimo, es fundamental comprender las bases teóricas que subyacen en las Máquinas de Vectores de Soporte y los principios matemáticos que permiten maximizar la separación entre las distintas clases. En la siguiente sección, profundizaremos en estos conceptos, explorando cómo se logra este objetivo.

3.2. Algorismo de las Máquinas de Soporte Vectorial

Introducción al algoritmo

Las **Máquinas de Soporte Vectorial** fueron desarrolladas por **Vladimir Vapnik** y **Alexey Chervonenkis** en 1963, como una extensión de la teoría estadística de la probabilidad y el aprendizaje automático. Su principal objetivo es encontrar un **hiperplano óptimo** que maximice el margen de separación entre dos clases en un espacio de características, lo que las convierte en una de las herramientas más populares para la clasificación binaria en el contexto del aprendizaje supervisado.

El principio fundamental de las SVM es que, para lograr una clasificación efectiva, la separación entre clases no debe ser simplemente un "corte" en el espacio, sino que debe maximizar la distancia entre las muestras más cercanas de cada clase, conocidas como **vectores de soporte**. Este enfoque no solo mejora la precisión de la clasificación, sino que también asegura una mayor capacidad de generalización, lo que permite que sean muy efectivas, incluso cuando se enfrentan a conjuntos de datos complejos o ruidosos.

Originalmente, las Máquinas de Vectores de Soporte fueron diseñadas para resolver problemas en los que los datos eran linealmente separables, es decir, en los que existía un hiperplano claro capaz de dividir las dos clases de manera exacta. Sin embargo, a medida que surgieron problemas más complejos, se descubrió que, en muchos casos, los

datos no podían separarse de forma lineal. Para abordar esta limitación, Vapnik introdujo el concepto de **márgenes suaves** (*soft margins*), una técnica que permite encontrar una solución óptima incluso cuando los datos no son perfectamente separables, tolerando ciertos errores de clasificación sin comprometer la precisión global del modelo. Además, se incorporó el uso de **núcleos** (*kernels*), que proyectan los datos en un espacio de mayor dimensión donde pueden separarse linealmente. A lo largo de este capítulo, se profundiza en la explicación de estos dos conceptos.

En esta sección, se explora cómo las SVM pueden aplicarse en problemas de clasificación binaria en tres casos distintos:

1. Datos **separables** linealmente.
2. Datos **casi separables** linealmente.
3. Datos **no separables** linealmente.

Estas categorías permiten comprender cómo las SVM abordan diferentes tipos de problemas y cómo los métodos de márgenes suaves y núcleos desempeñan un papel crucial en cada uno de estos casos.

3.2.1. Casos separables linealmente

Consideremos un conjunto de datos de entrenamiento que se representa de la siguiente forma:

$$S = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n},$$

donde $\mathbf{x}_i \in \mathbb{R}^d$ representa el vector de características de un punto de los datos i en un espacio d-dimensional, y $y_i \in \{-1, +1\}$ es la etiqueta asociada.

Para comprender cómo se construyen los clasificadores lineales en este contexto, comenzaremos definiendo los conceptos fundamentales.

Definición 3.2 (**Hiperplano**). *Un **hiperplano** en un espacio d-dimensional es un subespacio de una dimensión menos que su espacio ambiente. Puede representarse mediante una ecuación lineal de la forma:*

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0, \tag{3.2.1.1}$$

donde $\mathbf{w} \in \mathbb{R}^d$ es un vector normal al hiperplano, $\mathbf{x} \in \mathbb{R}^d$ es un punto en el espacio, y $b \in \mathbb{R}$ es un término de sesgo o desplazamiento. El hiperplano divide el espacio en dos regiones separadas según el signo de $\langle \mathbf{w}, \mathbf{x} \rangle + b$.

Para abordar problemas de clasificación, necesitamos introducir el concepto de un hiperplano de separación.

Definición 3.3 (**Hiperplano de separación**). *Un **hiperplano de separación** es una función lineal capaz de dividir un conjunto S sin error. Su ecuación es:*

$$D(\mathbf{x}_i) = \sum_{j=1}^{d} w_j x_{ij} + b = \langle \mathbf{w}, \mathbf{x}_i \rangle + b,$$

donde $\mathbf{w} \in \mathbb{R}^d$ y $b \in \mathbb{R}$.

Las restricciones que debe satisfacer un hiperplano de separación para cualquier dato de entrenamiento son:

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 0 \text{ si } y_i = +1,$$
$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq 0 \text{ si } y_i = -1.$$

Un conjunto de datos se denomina linealmente separable si se cumple la condición específica que definimos a continuación:

Definición 3.4 (**Linealmente separable**). *Diremos que* $S = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$ *es* ***linealmente separable*** *si existe un vector* $\mathbf{w} \in \mathbb{R}^d$ *y un escalar* $b \in \mathbb{R}$ *tal que:*

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0, \quad \forall i.$$

En este contexto, los vectores de soporte desempeñan un papel crucial, ya que son los puntos que definen el margen y el hiperplano óptimo.

Definición 3.5 (**Vectores de soporte**). *Los* ***vectores de soporte*** *son aquellos ejemplos de entrenamiento para los cuales* $\alpha_i > 0$. *Satisfacen:*

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1.$$

Estos puntos determinan el margen y el hiperplano óptimo. El vector normal $\mathbf{w}$ del hiperplano se obtiene como:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

El sesgo b se calcula a partir de cualquier vector de soporte:

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i.$$

El concepto de margen es fundamental en las Máquinas de Soporte Vectorial, ya que está directamente relacionado con la generalización del modelo.

Definición 3.6 (**Margen**). *Se define el* ***margen*** τ *como la distancia mínima entre el hiperplano de separación y los vectores de soporte, que son los puntos más cercanos al hiperplano en ambas clases. Geométricamente:*

$$\tau = \frac{1}{\|\mathbf{w}\|},$$

donde $\|\mathbf{w}\|$ *es la norma euclidiana del vector normal* $\mathbf{w}$.

Además, distinguimos entre el margen funcional y el margen geométrico.

Definición 3.7 (**Margen funcional**). *Dado un hiperplano definido por* $\mathbf{w}$ *(vector normal) y* b *(sesgo), el* ***margen funcional*** *para un ejemplo* $(\mathbf{x}_i, y_i)$ *se define como:*

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b).$$

Definición 3.8 (**Margen geométrico**). *El* ***margen geométrico*** *ajusta la escala para garantizar que sea independiente de la magnitud de* $\mathbf{w}$*:*

$$\frac{y_i(\mathbf{w} \cdot \mathbf{x}_i + b)}{\|\mathbf{w}\|}.$$

Como hemos mencionado, el objetivo es encontrar el **hiperplano óptimo** que mejor separa los datos. En contraste con el Perceptrón, que encuentra un hiperplano cualquiera, las SVM maximizan el **margen** entre las clases para garantizar una mejor generalización.

El margen máximo está directamente relacionado con los vectores de soporte, como se explica a continuación:

Proposición 3.9 (**Determinación del margen máximo**). *El margen máximo es determinado solo por los puntos más cercanos al conjunto de datos, conocidos como los* ***vectores de soporte****.*

Demostración. Los vectores de soporte son los puntos más cercanos al hiperplano de separación. Es decir, son los puntos para los cuales se cumple la siguiente ecuación:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1, \quad \forall i.$$

Estos puntos definen el límite entre las dos clases y son los responsables de determinar el **margen máximo**.

Por otro lado, los puntos que se encuentran alejados del hiperplano no afectan al cálculo del margen, ya que su contribución al margen es nula. El **margen máximo** se define exclusivamente por los vectores de soporte, ya que son los que están en el borde del margen y determinan la distancia entre el hiperplano y las clases.

Dado que el margen depende solo de los vectores de soporte, los puntos alejados no influyen en el valor del margen ni en la ubicación del hiperplano. Este hecho implica que el **margen máximo** está determinado únicamente por los vectores de soporte y no por otros puntos que puedan estar más alejados del hiperplano. □

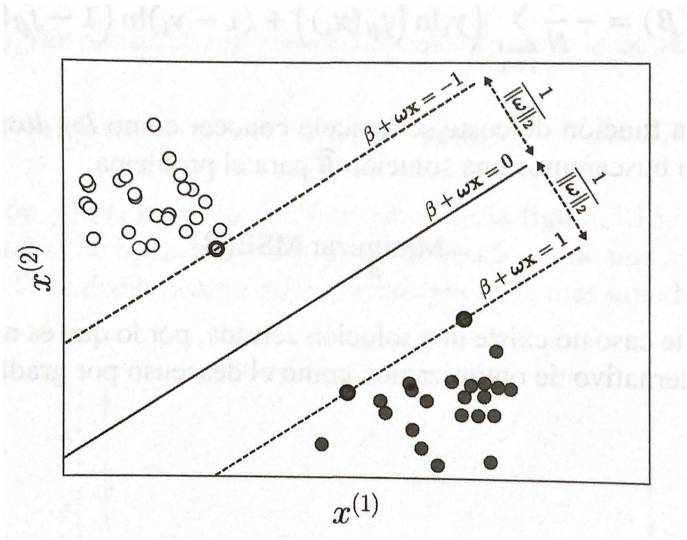

Figura 9: Clasificador óptimo para el problema de clasificación binaria de casos linealmente separables [3]

Finalmente, se dice que un hiperplano separador es **óptimo** si maximiza el margen τ. A continuación, se enuncia la condición que debe cumplir para ser considerado óptimo.

Proposición 3.10. *Un hiperplano separador es óptimo si y solo si equidista del ejemplo más cercano de cada clase.*

Demostración. Por reducción al absurdo, supongamos que la distancia del hiperplano óptimo al ejemplo más cercano de la clase $+1$ es menor que la distancia al ejemplo más cercano de la clase -1. En ese caso, sería posible mover el hiperplano más cerca del ejemplo de la clase -1 sin reducir el margen para la clase $+1$. Esto contradice la suposición de que el margen es máximo. Por lo tanto, el hiperplano óptimo debe equidistar de los puntos más cercanos de ambas clases. □

Formulación Matemática del Problema

La distancia entre un ejemplo $\mathbf{x}_i$ y el hiperplano definido por $\mathbf{w} \cdot \mathbf{x} + b = 0$ es:

$$\frac{|\mathbf{w} \cdot \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

De esta definición se deduce que maximizar el margen τ es equivalente a minimizar $\|\mathbf{w}\|$. Sin embargo, existen infinitas formas de definir un mismo hiperplano que difieren solo en la escala de $\mathbf{w}$. Por ejemplo, todas las funciones lineales $\varphi(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$, con $\varphi \in \mathbb{R}$, representan el mismo hiperplano. Para limitar, por tanto, el número de soluciones a una sola, la escala del producto φ y la norma de $\mathbf{w}$ se fija de forma arbitraria a la unidad, es decir:

$$\|\mathbf{w}\| = 1$$

Para garantizar que todos los ejemplos de entrenamiento estén correctamente clasificados y fuera del margen, se impone la siguiente restricción:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n$$

Este es el sistema de restricciones que debe cumplirse para todo i. El problema de encontrar el hiperplano óptimo para el caso de clasificación binaria de ejemplos linealmente separables puede formalizarse como el problema de encontrar los valores $\mathbf{w}$ y b que minimizan el funcional $f(\mathbf{w}) = \|\mathbf{w}\|$, sujeto a las restricciones dadas por la ecuación anterior. De manera equivalente, esto se expresa como:

$$\begin{aligned} & \text{minimizar}_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{sujeto a:} \quad & y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1, \quad i = 1, \ldots, n \end{aligned}$$

Problema Dual

Como se mencionó en la sección anterior, un problema de minimación primal tiene una forma dual si la función a minimizar y las restricciones son funciones estrictamente convexas. En este caso, el problema primal cumple con el criterio de convexidad, por lo que tiene un problema dual. Aplicando los resultados descritos, podemos transformar el problema primal en su dual.

Primero, construimos la **función Lagrangiana** $L(\mathbf{w}, b, \alpha)$:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \right]$$

donde $\alpha_i \geq 0$ son los multiplicadores de Lagrange.

Condiciones de Karush-Kuhn-Tucker (KKT)

A continuación, aplicamos las condiciones KKT. Derivando la función Lagrangiana con respecto a $\mathbf{w}$, b, y α_i, obtenemos:

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{3.2.1.2}$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.2.1.3}$$

$$\alpha_i \left[1 - y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)\right] = 0, \quad i = 1, \ldots, n \tag{3.2.1.4}$$

Las dos primeras restricciones (3.2.1.2), (3.2.1.3) corresponden a la primera condición KKT, mientras que la última (3.2.1.4) es la condición complementaria.

Construcción del Problema Dual

Para construir el problema dual, primero sustituimos la expresión de $\mathbf{w}$ en la función Lagrangiana. Reescribimos $L(\mathbf{w}, b, \alpha)$ como:

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

Este es el problema dual, que consiste en maximizar la función $L(\alpha)$ sujeto a las restricciones:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, \ldots, n$$

Solución del Problema Dual

La solución del problema dual permite obtener la solución del problema primal. Para ello, basta con sustituir la solución obtenida en la expresión de $\mathbf{w}$ y, finalmente, sustituir el resultado en la ecuación del hiperplano:

$$D(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b$$

Para determinar el valor de b, se hace uso de las restricciones expresadas en (3.2.4), de manera que, si $\alpha_i > 0$, entonces el segundo factor de la parte izquierda de las restricciones debe ser cero. Por lo tanto, se cumple que:

$$y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) = 1$$

Por definición, los ejemplos que satisfacen esta ecuación son los vectores soporte, es decir, aquellos que tienen asociado un $\alpha_i > 0$. Esto nos lleva a la conclusión de que el hiperplano de separación se construye únicamente como una combinación lineal de los vectores soporte (Prop 3.9).

Finalmente, el valor de b se obtiene despejándolo de la ecuación anterior:

$$b = y_{\text{vs}} - \langle \mathbf{w}, \mathbf{x}_{\text{vs}} \rangle$$

donde $(x_{\text{vs}}, y_{\text{vs}})$ es cualquier vector soporte. En la práctica, es más robusto obtener el valor de b promediando los resultados de todos los vectores soporte:

$$b = \frac{1}{N_{\text{VS}}} \sum_{j \in \text{VS}} (y_j - \langle \mathbf{w}, \mathbf{x}_j \rangle)$$

donde N_{VS} es la cantidad de vectores soporte.

3.2.2. Casos cuasi-separables linealmente

En problemas reales, es común encontrarse con conjuntos de datos que no son perfectamente separables de manera lineal. Para abordar este desafío, se ha demostrado que es más efectivo permitir que algunos ejemplos no cumplan estrictamente las restricciones del margen formuladas en el problema primal.

Para manejar esta situación, se introducen las **variables de holgura** (ξ_i) y un **parámetro** C que controla el equilibrio entre los errores de clasificación y el ancho del margen. El problema a resolver se plantea como:

$$\text{minimizar}_{\mathbf{w},b,\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i,$$
$$\text{sujeto a:} \quad y_i(\mathbf{w} \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n.$$

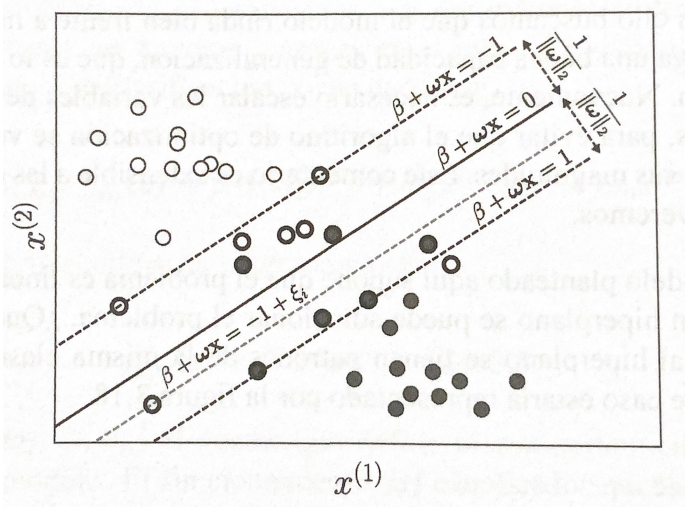

Figura 10: Clasificador óptimo para el problema de clasificación binaria de casos linealmente cuasi-separables [3]

Las variables de holgura se interpretan como desviaciones de la condición de separabilidad lineal. Si $\xi_i = 0$, el ejemplo (x_i, y_i) es separable; si $0 < \xi_i \leq 1$, el ejemplo es correctamente clasificado pero no estrictamente separable; y si $\xi_i > 1$, el ejemplo no es

separable y está mal clasificado. Por tanto, el valor total $\sum_{i=1}^{n} \xi_i$ refleja el costo asociado al número de ejemplos no separables: a mayor suma, mayor será el número de ejemplos mal clasificados.

El objetivo es minimizar la siguiente función:

$$f(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i,$$

donde ξ_i representa los errores de clasificación y C es el parámetro que controla la penalización de los errores. La elección de un valor de C determinará la calidad del clasificador, ya que un valor más alto de C reducirá los errores de clasificación a costa de un margen más pequeño, mientras que un valor bajo de C permitirá un margen mayor a cambio de más errores de clasificación.

El hiperplano que resuelve este último problema se denomina **hiperplano de separación de margen blando**.

Transformación a la forma dual

Como en el caso de la sección anterior, el problema de optimización original (problema primal) es transformado a su forma dual. El procedimiento para obtener el hiperplano de separación es similar al utilizado en el caso de datos separables linealmente. Por tanto, aquí sólo se reproduce de forma esquemática y secuencial los pasos necesarios para realizar dicha transformación.

Paso 1: Obtención de la función lagrangiana. La función lagrangiana del problema primal se expresa como:

$$L(\mathbf{w}; b; \alpha; \beta) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C\sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \beta_i \left[y_i \left(\langle \mathbf{w}, x_i \rangle + b \right) + \alpha_i - 1 \right]$$

Paso 2: Aplicación de las condiciones KKT. Las condiciones KKT para la función lagrangiana son las siguientes:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \tag{3.2.2.1}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.2.2.2}$$

$$\frac{\partial L}{\partial \alpha_i} = C - \alpha_i - \beta_i = 0, \quad i = 1, \ldots, n \tag{3.2.2.3}$$

$$\beta_i \left[1 - y_i \left(\langle \mathbf{w}, x_i \rangle + b \right) - \alpha_i \right] = 0, \quad i = 1, \ldots, n$$

$$\beta_i \alpha_i = 0, \quad i = 1, \ldots, n \tag{3.2.2.4}$$

Paso 3: Establecimiento de relaciones entre las variables. Las relaciones entre las variables del problema primal (w, b, α) y las del problema dual (α, β) se obtienen a partir de la condición (3.2.2.1):

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{3.2.2.5}$$

Paso 4: Establecimiento de restricciones adicionales para las variables duales. Para las variables duales, se establecen las siguientes restricciones usando las relaciones (3.2.2.2) y (3.2.2.3):

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$
$$C = \alpha_i + \beta_i, \quad i = 1, \ldots, n$$

Paso 5: Eliminación de las variables primales. Eliminando las variables primales de la función lagrangiana, se obtiene el problema dual que se desea maximizar:

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

La formalización del problema dual es la siguiente:

$$\text{maximizar}_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$
$$\text{sujeto a:} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \tag{3.2.2.6}$$

Solución del Problema Dual

Como en el caso anterior, la solución del problema dual nos permite expresar el hiperplano de separación óptimo en términos de α_i. Para ello, basta con sustituir la expresión (3.2.2.5) en la ecuación (3.2.1.1):

$$D(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x, x_i \rangle + b$$

Antes de obtener una expresión precisa para b, se consideran algunos resultados interesantes:

- De la restricción (3.2.2.6), se deduce que si $\alpha_i = 0$, entonces $C = \beta_i$. Esto, junto con la restricción (3.2.2.4), implica que $\alpha_i = 0$. Por lo tanto, todos los ejemplos x_i cuyo α_i sean igual a cero corresponden a ejemplos separables.

- Por otro lado, todo ejemplo no separable, x_i, se caracteriza por tener asociado un $\alpha_i > 0$. En este caso, y de la restricción (3.2.2.4), se deduce que $\alpha_i = C$, lo que corresponde a ejemplos no separables.

El valor de b se puede calcular mediante la siguiente fórmula:

$$b = y_i - \langle \mathbf{w}, x_i \rangle \quad \forall i \text{ tal que } 0 < \alpha_i < C$$

En este caso, para el cálculo de b, no es suficiente con elegir cualquier ejemplo x_i con $\alpha_i > 0$; se requiere que α_i esté en el intervalo $0 < \alpha_i < C$.

También se puede obtener una estimación más precisa de b promediando los valores obtenidos para todos los ejemplos que cumplen con $0 < \alpha_i < C$:

$$b = \frac{1}{|V_{0S}|} \sum_{j \in V_{0S}} (y_j - \langle \mathbf{w}, x_j \rangle)$$

donde V_{0S} es el conjunto de ejemplos con α_i tal que $0 < \alpha_i < C$, y $|V_{0S}|$ es la cardinalidad de este conjunto.

Finalmente, también es posible expresar b en términos de las variables duales usando la expresión:

$$b = y_i - \sum_{j=1}^{n} \alpha_j y_j \langle x_j, x_i \rangle \quad \forall \alpha_i \text{ tal que } 0 < \alpha_i < C$$

donde los coeficientes α_i, con $i = 1, \ldots, n$, corresponden a la solución del problema dual.

3.2.3. Casos no separables linealmente

Cuando un conjunto de ejemplos no puede ser separado por un hiperplano lineal, utilizamos una técnica alternativa que consiste en transformar el espacio original mediante una función no lineal, llevando los datos a un espacio de Hilbert [17]. Este espacio está dotado de un producto escalar y se denomina **función Núcleo**.

El **espacio de entradas** es el espacio donde se encuentran los datos originales o las observaciones que se quieren clasificar. Este espacio está formado por los vectores $\mathbf{x}_i \in \mathbb{R}^d$, donde d es la dimensión de cada ejemplo de entrada. Los puntos en este espacio corresponden a las características de los ejemplos de entrenamiento en su forma original, antes de ser transformados por la función Núcleo.

Definición 3.11. *El **espacio de entradas** es el espacio donde se encuentran los datos originales o las observaciones que se quieren clasificar. Este espacio está formado por los vectores $\mathbf{x}_i \in \mathbb{R}^d$, donde d es la dimensión de cada ejemplo de entrada. Los puntos en este espacio corresponden a las características de los ejemplos de entrenamiento en su forma original, antes de ser transformados por la función Núcleo.*

Sin embargo, cuando los datos no son separables linealmente en el espacio de entradas, los mapeamos a un **espacio de características** mediante una función de transformación no lineal $\phi(\mathbf{x})$, que puede ser de alta o infinita dimensión. En este nuevo espacio, los datos pueden volverse separables linealmente, lo que permite encontrar un hiperplano que los separe de manera adecuada.

Definición 3.12. *El **espacio de características** es el espacio al que los datos son mapeados mediante la función de transformación no lineal $\phi(\mathbf{x})$, que puede ser de alta o infinita dimensión. En este espacio, los datos pueden volverse separables linealmente, lo que permite encontrar un hiperplano que los separe de manera adecuada.*

Para evitar trabajar directamente con la transformación $\phi(\mathbf{x})$, podemos utilizar la función núcleo, que es mucho más eficiente. La **función núcleo** o *función kernel* está asociada al producto escalar en el espacio de características $\mathcal{F}$.

Definición 3.13. *Definimos la función núcleo o función kernel como el producto escalar en el espacio de características $\mathcal{F}$ asociado a la función de transformación ϕ. Formalmente, la función kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ es la función que calcula el producto escalar entre las imágenes de los puntos $\mathbf{x}_i$ y $\mathbf{x}_j$ bajo la transformación ϕ:*

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{F}}.$$

A continuación, para comprender mejor el concepto, se presenta el siguiente ejemplo con su ilustración. Para ello, se parte de la transformación $\varphi((a,b)) = (a, b, a^2 + b^2)$, que mapea puntos bidimensionales a un espacio tridimensional. En este espacio, el producto interno se define como:

$$\varphi((a,b)) \cdot \varphi((c,d)) = a \cdot c + b \cdot d + (a^2 + b^2)(c^2 + d^2).$$

Aplicando la propiedad distributiva, el término cuadrático $(a^2 + b^2)(c^2 + d^2)$ se reescribe utilizando las normas $\|\mathbf{x}\|^2 = a^2 + b^2$ y $\|\mathbf{y}\|^2 = c^2 + d^2$. Así, el producto interno queda expresado como:

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} + \|\mathbf{x}\|^2 \|\mathbf{y}\|^2,$$

lo que define el kernel directamente en el espacio original.

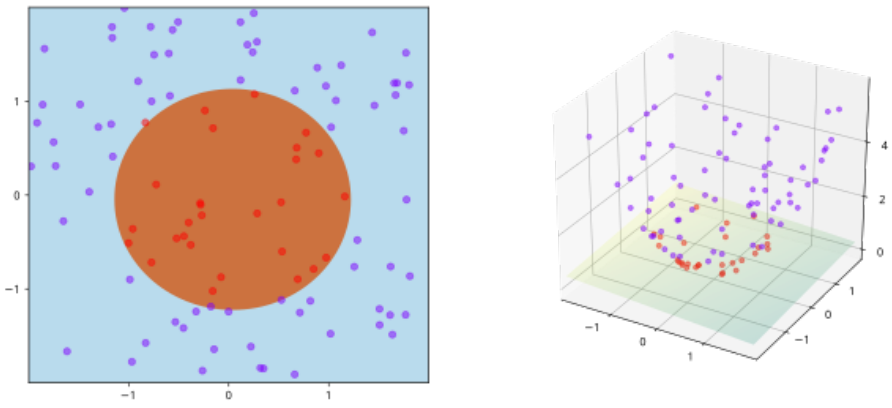

Figura 11: Ejemplo de la aplicación anterior en el espacio de entradas para obtener el de características donde se puede encontrar fácilmente un hiperplano de separación [16]

Para acabar de entender el concepto se proporciona otra imagen. Esta muestra la clasificación realizada por una Máquina de Soporte Vectorial utilizando el kernel RBF (Radial Basis Function) con diferentes valores de γ. En ella, se representan las zonas asignadas por el modelo a cada clase, destacando cómo la elección de γ afecta la forma y la separación de las regiones de decisión.

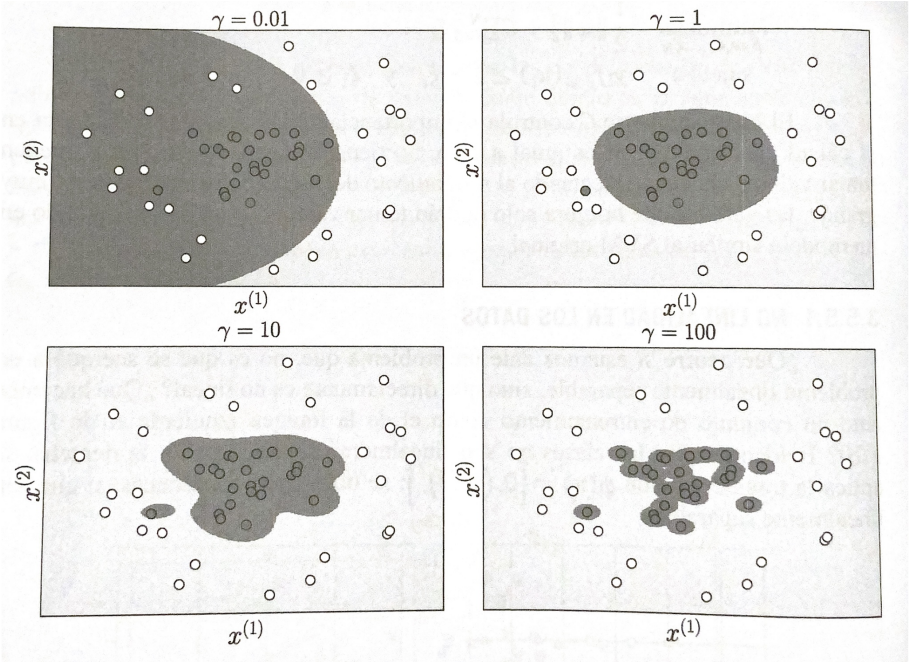

Figura 12: SVM utilizando el kernel RBF con varios valores de γ. Se representan las zonas asignadas por el modelo a cada clase [3].

A continuación, se presentan algunos ejemplos comunes de funciones núcleo utilizadas en SVM:

Núcleo lineal:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j.$$

Núcleo polinómico:

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d,$$

Núcleo sigmoide:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\mathbf{x}_i \cdot \mathbf{x}_j + c).$$

El teorema siguiente, el **Teorema de Aronszajn**, es fundamental en el análisis funcional y en el estudio de espacios de Hilbert, ya que establece la existencia y unicidad de un núcleo asociado a cualquier espacio de características dado. Este resultado tiene aplicaciones significativas en el campo del aprendizaje automático, especialmente en el contexto de los métodos de máquinas de vectores de soporte, donde los espacios de características y los núcleos juegan un papel crucial en la clasificación y el análisis de datos.

La importancia del Teorema de Aronszajn radica en su capacidad para garantizar que, dados ciertos datos o características, siempre existe una función que puede transformar el espacio original de entrada a un espacio de Hilbert donde se pueden aplicar técnicas lineales de manera eficiente. De este modo, el teorema facilita el trabajo con espacios no lineales mediante el uso de funciones núcleo, lo que permite un enfoque más general y robusto para la resolución de problemas complejos en el aprendizaje automático.

Teorema 3.14 (Teorema de Aronszajn)**.** *Sea $\mathcal{H}$ un espacio de Hilbert. Entonces, para cualquier espacio de características $\mathcal{F}$, existe un único núcleo $k(\mathbf{x}_i, \mathbf{x}_j)$ asociado a este espacio de características.*

Volvemos a la formulación del problema no separable linealmente. La idea es construir un hiperplano de separación lineal en el subespacio de características. La frontera de decisión lineal obtenida en el nuevo subespacio se transformará en la frontera de decisión no lineal en el espacio original de entradas. En este contexto, el hiperplano separador en el espacio de características viene dado por la siguiente ecuación:

$$\sum_{i=1}^{n} \mathbf{w_i}\phi_i(\mathbf{x}) + b = 0$$

donde $\phi(\mathbf{x})$ es la función de transformación que mapea los datos del espacio original de entradas $\mathbb{X}$ al espacio de características $\mathcal{F}$.

El planteamiento del problema es el mismo que en la sección anterior, con la diferencia de que ahora sustituimos el problema dual por la función núcleo. Es decir, en lugar de trabajar directamente con el espacio de características, utilizamos un núcleo para calcular el producto escalar de los puntos en este espacio sin necesidad de realizar la transformación explícita. El **problema dual** en este caso se puede escribir de la siguiente forma:

$$\text{maximizar}_{\alpha} \quad \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \right),$$

$$\text{sujeto a: } \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n,$$

donde $k(\mathbf{x}_i, \mathbf{x}_j)$ es la función núcleo que sustituye el producto escalar en el espacio de características $\mathcal{F}$.

Gracias al teorema siguiente, el **teorema del Representante** , sabemos que la solución óptima puede expresarse únicamente en términos de los puntos de entrenamiento $x_1, x_2, \ldots, x_n$, y los coeficientes $\alpha_1, \alpha_2, \ldots, \alpha_n$ obtenidos durante el proceso de optimización. Esto simplifica considerablemente el cálculo, ya que no necesitamos trabajar directamente en el espacio de características, sino que podemos realizar todos los cálculos necesarios utilizando el núcleo.

Ambos teoremas, el de Aronszajn y el de Representante, están estrechamente relacionados. El primero establece la existencia de un núcleo para cualquier espacio de características, mientras que el segundo nos dice cómo la solución óptima puede expresarse utilizando ese núcleo. Juntos, estos resultados teóricos nos permiten realizar la optimización en el espacio de características sin necesidad de transformaciones explícitas, lo que hace posible el uso eficiente de SVM y otros métodos basados en núcleos.

Teorema 3.15 (Teorema del representante)**.** *Queremos encontrar f^*, que es la solución del siguiente problema de minimización:*

$$f^* = \arg \mathit{minimizacion}_{f \in H} \left(\frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) + \lambda \|f\|_H^2 \right),$$

donde $L(f(x_i), y_i)$ es la función de error de clasificación y $\|f\|_H^2$ es la norma de la función f en el espacio de Hilbert H, que es el RKHS (Reproducing Kernel Hilbert Space) asociado al núcleo k.

La solución óptima del problema anterior está dada por:

$$f^*(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x),$$

donde $\alpha_i \in \mathbb{R}$ son los coeficientes resultantes de la optimización y $k(x_i, x)$ es el valor del núcleo evaluado en los puntos de entrenamiento x_i y el punto x de entrada.

Demostración. Sea H el espacio de Hilbert generado por todas las combinaciones lineales de las funciones núcleo evaluadas en los puntos de entrenamiento $x_1, x_2, \ldots, x_n$, es decir:

$$S = \operatorname{span}\{k(x_1, \cdot), k(x_2, \cdot), \ldots, k(x_n, \cdot)\}.$$

Esto significa que el espacio H está generado por los núcleos de los puntos de entrenamiento, y que cualquier función $f \in H$ puede escribirse como una combinación lineal de estos núcleos.

La propiedad de reproducción de los núcleos implica que para cualquier x_i, el valor de f en x_i está dado por el producto interno con el núcleo correspondiente:

$$f(x_i) = \langle f, k(x_i, \cdot) \rangle_H.$$

Podemos descomponer el espacio de hipótesis H en dos componentes:

$$f = f_S + f_K,$$

donde $f_S \in S$ y f_K es ortogonal a S, es decir, $f_K \perp S$.

Como estamos buscando f^*, que minimiza la función de costo, debemos considerar el término de la norma $\|f\|_H^2$. Dado que f_K es ortogonal a S, podemos escribir la función f^* en términos de S (es decir, en el espacio generado por los núcleos evaluados en los puntos de entrenamiento).

Así, la solución f^* debe estar en S, es decir, $f^* \in S$.

El objetivo es minimizar:

$$f^* = \arg\min_{f \in S} \left(\frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) + \lambda \|f\|_H^2 \right).$$

Como $f^* \in S$, podemos escribir f^* como una combinación lineal de los núcleos:

$$f^*(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x),$$

donde $\alpha_i \in \mathbb{R}$ son los coeficientes resultantes de la optimización.

Por lo tanto, la solución óptima f^* se puede expresar como:

$$f^*(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x),$$

donde α_i son los coeficientes resultantes de la optimización y $k(x_i, x)$ es el valor del núcleo evaluado en los puntos de entrenamiento x_i y el punto de entrada x. □

En este contexto, la solución óptima para el clasificador en el espacio RKHS se puede expresar como una combinación lineal de los puntos de entrenamiento. Esto implica que la función de decisión se construye utilizando los coeficientes α_i, ajustados durante el proceso de optimización, y las similitudes entre los puntos de entrada, calculadas a través de una función núcleo. Si se relaciona el teorema al caso de estudio se tiene que:

- $L(f(x_i), y_i)$: Es la función de pérdida, que mide el error del modelo en el conjunto de entrenamiento.
- λ: Es el parámetro de regularización, que controla el equilibrio entre minimizar el error en el conjunto de entrenamiento y la complejidad del modelo.
- $\|f\|_H^2$: Es la norma de la función f en el espacio RKHS asociado al núcleo k.

Este enfoque tiene implicaciones significativas. Por un lado, permite manejar problemas no lineales de clasificación al proyectar los datos en un espacio de características de mayor dimensión, definido implícitamente por el núcleo. Por otro lado, gracias a la regularización incluida en el modelo ($\lambda\|f\|_H^2$), se garantiza que la solución dependa de un conjunto relativamente pequeño de puntos de entrenamiento llamados vectores de soporte. Estos puntos son críticos para definir el clasificador, lo que da lugar a modelos robustos con alta capacidad de generalización.

Una vez comprendidos los conceptos básicos necesarios para entender el funcionamiento de las Máquinas de Vectores de Soporte en problemas binarios, en el siguiente capítulo se presenta un **caso práctico**. En ese ejercicio, se explora la aplicación del modelo en un escenario real para resolver un problema de clasificación. A través de este ejemplo, se muestra cómo entrenar y evaluar un modelo SVM utilizando un conjunto de datos etiquetado. Además, se abordan aspectos clave del proceso, como la selección del kernel, la optimización de parámetros y la interpretación de los resultados obtenidos.

4. Caso Práctico: Aplicación de las SVM

4.1. Introducción

El **análisis de sentimientos**, también conocido como minería de opiniones, es una técnica del **procesamiento de lenguaje natural** (PLN)[8], análisis de texto y lingüística computacional que permite identificar, extraer y clasificar automáticamente las emociones, percepciones o actitudes subjetivas presentes en los textos digitales. Esta técnica se ha convertido en una herramienta esencial en sectores como el *marketing*, la monitorización de redes sociales y la investigación de mercados, al ofrecer una manera eficiente de interpretar grandes volúmenes de opiniones expresadas en diversos contextos.

El objetivo principal de este proyecto ha sido desarrollar un modelo de inteligencia artificial basado en el algoritmo de **Máquinas de Vectores de Soporte** para clasificar textos según su polaridad: positiva o negativa. Este trabajo se enmarca dentro del análisis de sentimientos, una disciplina cuyo propósito es identificar y extraer las opiniones o emociones expresadas en un texto.

El sistema SVM ha sido entrenado utilizando un conjunto de datos etiquetados, donde a cada texto se le asigna una etiqueta que indica su polaridad. A partir de este modelo, se ha evaluado un conjunto de opiniones extraídas de Amazon y otras fuentes relacionadas con teléfonos móviles de diversas marcas. Una vez entrenado y evaluado el modelo con la primera base de datos, se procede a evaluar una segunda base de datos que ya cuenta con etiquetas, pero sin proporcionarlas al modelo durante el proceso. Posteriormente, se comparan las predicciones del modelo con las etiquetas reales para determinar cuántas de ellas coinciden. Este paso nos permite observar cuán bien se ajusta el modelo a la realidad.

La parte práctica de este proyecto ha sido diseñada para aplicar de manera concreta el algoritmo de clasificación estudiado. Para ello, se ha optado por una aplicación sencilla pero efectiva: la clasificación de críticas según su polaridad. Esta elección no solo ha permitido observar los resultados del modelo, sino también validar su capacidad para procesar y analizar grandes volúmenes de datos textuales provenientes de plataformas, lo cual es un desafío clave en el campo del análisis de sentimientos.

Subtareas del Análisis de Sentimiento

El análisis de sentimiento abarca varias subtareas, una de ellas es la **clasificación de polaridad**[9], que consiste en determinar si el contenido de un texto expresa una opinión positiva, negativa o neutral.

Otra subtarea es la **detección de emociones**, que busca identificar sentimientos específicos como felicidad, tristeza o enojo.

Finalmente, se incluye la **identificación subjetiva/objetiva**, que clasifica un texto como subjetivo, es decir, expresiones de opinión, u objetivo, cuando está enfocado en transmitir información neutral.

[8] El procesamiento de lenguaje natural (PLN) es un subcampo de la informática y la inteligencia artificial (IA) que utiliza técnicas de *machine learning* para permitir que las computadoras entiendan y se comuniquen en lenguaje humano.

[9] La polaridad se refiere a la orientación emocional de un texto, es decir, si expresa una opinión positiva, negativa o neutral.

Aplicaciones del Análisis de Sentimiento

El **análisis de sentimientos** tiene diversas aplicaciones prácticas en áreas como el monitoreo de redes sociales para comprender las opiniones públicas, la mejora de productos mediante el análisis de críticas y comentarios de los usuarios, así como la predicción de tendencias políticas y económicas.

No obstante, este campo enfrenta importantes desafíos, entre los que destacan: el **sarcasmo** y la **ironía**, que resultan difíciles de interpretar para los sistemas automatizados; el **contexto** y la **ambigüedad**, ya que muchas palabras y expresiones adquieren diferentes significados dependiendo del contexto lingüístico en el que se utilicen; y las **diferencias culturales y lingüísticas**, que complican la generalización de los modelos debido a las variaciones idiomáticas presentes en diferentes regiones y lenguajes.

El proceso de análisis de sentimientos con inteligencia artificial involucra una serie de pasos estructurados que comienzan con la recolección de datos y culminan con el análisis realizado por los modelos de aprendizaje automático. A continuación, se explica cada etapa de manera teórica y la aplicación de esta en detalle.

4.2. Metodología

Recolección de los datos

El primer paso es la **recolección de datos**. En este punto, se obtiene el texto que será analizado. Los datos pueden provenir de diversas fuentes como redes sociales, foros de discusión, reseñas de productos, artículos de noticias, blogs, entre otros. Dependiendo del tipo de análisis que se desee realizar, se seleccionan las fuentes más adecuadas. En esta fase, es importante asegurarse de que los datos sean relevantes, de alta calidad y representativos del contexto que se desea estudiar.

Para obtener un conjunto de datos que refleja mejor la información presente en las reseñas, se seleccionó una base de datos con opiniones relacionadas con móviles, acompañadas de un calificador numérico en el rango [1, 5].

Cada texto tiene asociado un valor numérico representando el grado de positividad o negatividad de la opinión. Para simplificar la clasificación de polaridad, se agrupan las puntuaciones entre 1 y 2 como **negativas** y entre 4 y 5 como **positivas**. Las puntuaciones intermedias son eliminadas, ya que no se consideraron los casos **neutros** en este análisis.

Cada instancia de la base de datos incluye las siguientes propiedades:

- ***Text***: Un fragmento que representa una opinión, reseña o comentario.
- ***Target***: Número entre [1,5] según la calificación del texto.
- ***Sentiment***: Una clasificación binaria (**positiva** o **negativa**) asociada al texto.
- ***db***: Fuente de datos.

Text	Label	Sentiment	db
Este móvil es increíble.	4.5	Positivo	Samsung
La pantalla era muy grande.	4.2	Positivo	Apple
La resolución de la pantalla es baja.	2.1	Negativo	Apple

Cuadro 2: Ejemplo de una muestra de la base de datos resultante.

Preprocesamiento de los datos

Una vez se tiene la base de datos, se inicia el proceso de **limpieza de texto**, que es un paso fundamental para preparar los datos para el análisis posterior. Este proceso consiste en eliminar los **elementos ruidosos** e **irrelevantes del texto**, tales como caracteres especiales, que son símbolos extraños que no aportan valor al análisis; URLs, que son enlaces web que no contienen información útil para el análisis de sentimientos; y emojis, que aunque pueden expresar emociones, se eliminan si no se desea analizarlos específicamente.

Además, se realiza un proceso de **normalización**, que implica convertir todas las palabras a un formato estándar (por ejemplo, convirtiendo todo a minúsculas) y eliminar duplicados. La limpieza mejora la calidad de los datos y facilita los pasos posteriores de análisis.

El preprocesamiento de datos es una etapa crucial en el desarrollo del modelo, ya que los textos en bruto pueden contener elementos irrelevantes que podrían afectar negativamente al desempeño del algoritmo. A continuación, se describen los pasos clave en este proceso:

Limpieza del Texto

Durante la limpieza del texto, se aplican varias técnicas para reducir el ruido y asegurar que solo se mantenga la información relevante:

Eliminación de caracteres innecesarios: Se eliminan caracteres especiales, enlaces, emojis y números que no aportan información semántica relevante. Por ejemplo, cualquier símbolo que no sea una palabra significativa se elimina para mejorar la calidad del texto.

Conversión a minúsculas: Se convierten todas las palabras a minúsculas para garantizar que el algoritmo no trate palabras como "Pantalla" y "pantalla" como diferentes. Esto ayuda a mantener la uniformidad en el análisis.

Tokenización: El texto se divide en *tokens*, es decir, en palabras individuales que se utilizan para el análisis posterior. Esto convierte las frases en listas de palabras procesables por el modelo.

Eliminación de *stopwords*: Las *stopwords*[10] son palabras comunes, como artículos, preposiciones y conjunciones, que no contribuyen significativamente al análisis semántico del texto. Se eliminan para reducir la complejidad del modelo. Por ejemplo, palabras como "la", "y", "es" no aportan al significado del texto y se descartan. A su vez, se consideran también palabras comunes en plataformas como "rt", "via", "us", "u", etc.

Normalización: La normalización ajusta las palabras para que tengan una forma más estándar, eliminando acentos y otros elementos que podrían variar pero que no son relevantes para el análisis semántico. Por ejemplo, las palabras "matricula" y "matrícula" se considerarán iguales tras la normalización.

[10]Las *stopwords* o palabras vacías hacen referencia a aquellas palabras que carecen de sentido cuando se escriben solas o sin la palabra clave.

Lematización: Finalmente, la lematización reduce las palabras a su forma base, eliminando variaciones de género, número o tiempo verbal. Por ejemplo, la palabra "buena" se lematiza a "bueno", y "fue" a "ser". Esto ayuda a unificar las diferentes formas de una misma palabra.

Dado la siguiente reseña, relacionada con la satisfacción del cliente en un servicio de atención telefónica, se procede a hacer tal conversión:

"El servicio de atención al cliente fue excepcional, resolvieron mi problema rápidamente y fueron muy amables durante toda la llamada."

Se aplica el preprocesamiento paso a paso:

Eliminación de caracteres innecesarios: En este caso, no hay caracteres innecesarios como emojis o enlaces, por lo que la frase permanece igual.

Conversión a minúsculas: La frase convertida a minúsculas sería:

- "el servicio de atención al cliente fue excepcional, resolvieron mi problema rápidamente y fueron muy amables durante toda la llamada."

Tokenización: La frase se divide en *tokens* o palabras individuales. Los *tokens* generados serían:

- **Tokens generados**: ["el", "servicio", "de", "atención", "al", "cliente", "fue", "excepcional", "resolvieron", "mi", "problema", "rápidamente", "y", "fueron", "muy", "amables", "durante", "toda", "la", "llamada"]

Eliminación de *stopwords*: Se eliminan las palabras comunes que no aportan al análisis, como "el", "de", "al", "y", "mi", "la". Los vectores después de eliminar las *stopwords* serían:

- **Tokens después de eliminar stopwords**: ["servicio", "atención", "cliente", "fue", "excepcional", "resolvieron", "problema", "rápidamente", "fueron", "amables", "durante", "llamada"]

Normalización: Se normalizan las palabras eliminando tildes u otros signos ortográficos innecesarios. Los vectores normalizados serían:

- **Tokens normalizados**: ["servicio", "atencion", "cliente", "fue", "excepcional", "resolvieron", "problema", "rapidamente", "fueron", "amables", "durante", "llamada"]

Lematización: Las palabras se lematizan a su forma base. En este caso, "fue" se lematiza a "ser" y "resolvieron" a "resolver". Los vectores lematizados quedarían:

- **Tokens lematizados**: ["servicio", "atencion", "cliente", "ser", "excepcional", "resolver", "problema", "rapido", "ser", "amable", "durante", "llamar"]

Este proceso de preprocesamiento transforma el texto original en una forma más adecuada para ser analizada y procesada por un modelo de análisis de sentimientos.

Balance entre clases

En la base de datos creada se ha encontrado un desequilibrio entre la cantidad de datos negativos y positivos. Para evitar que el aprendizaje de la máquina se vea afectado por un sesgo, se ha seleccionado de manera aleatoria un número igual de ejemplos de ambos grupos. En este caso, se han encontrado más datos positivos que negativos. Por ello, se han seleccionado de manera aleatoria el mismo número para ambos, aproximadamente unas 77.000 muestras para cada clase.

Representación del Texto

Como los algoritmos SVM no pueden procesar texto directamente, ha sido necesario convertirlo en un formato numérico. El **TF-IDF** convierte los textos en vectores numéricos, midiendo la relevancia de cada palabra en un documento dentro de un corpus a partir de la fórmula:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t)$$

donde:

- $\text{TF}(t, d)$: Frecuencia de la palabra t en el documento d.
- $\text{IDF}(t) = \log(\frac{N}{1+\text{DF}(t)})$: Mide la importancia de t en el corpus, siendo N el número total de documentos y $\text{DF}(t)$ el número de documentos que contienen t.

A continuación, se detallan los pasos aplicados a un ejemplo con dos vectores:

$$[\text{“mobil”}, \text{“ser”}, \text{“increíble”}], \quad [\text{“pantalla”}, \text{“ser”}, \text{“muy”}, \text{“grande”}]$$

En este caso solamente analizaremos al detalle la primera, y de la segunda proporcionaremos únicamente el resultado final ya que se calcula de manera análoga.

Paso 1: Calcular TF para cada palabra

Palabra	**Frecuencia (TF)**
mobil	$\frac{1}{3}$
ser	$\frac{1}{3}$
increíble	$\frac{1}{3}$
pantalla	$\frac{0}{3} = 0$
muy	$\frac{0}{3} = 0$
grande	$\frac{0}{3} = 0$

Cuadro 3: Cálculo del TF para cada palabra del vector 1.

Paso 2 y Paso 3: Calcular IDF para cada palabra y Calcular TF-IDF para cada palabra

Fórmula para obtener la IDF:

$$\text{IDF}(t) = \log(\frac{N}{1 + \text{DF}(t)})$$

donde $N = 2$ (total de documentos) y $\mathrm{DF}(t)$ es el número de documentos que contienen la palabra t.

Y después, multiplicamos **TF** por **IDF** para obtener el valor de **TF-IDF** para cada palabra.

Palabra	**DF (Documento Frecuencia)**	**IDF Cálculo**	**Valor**
mobil	1	$\log(\frac{2}{1+1}) = \log(1) = 0$	0
ser	2	$\log(\frac{2}{1+2}) = \log(\frac{2}{3}) \approx -0{,}176$	-0.176
increíble	1	$\log(\frac{2}{1+1}) = \log(1) = 0$	0
pantalla	1	$\log(\frac{2}{1+1}) = \log(1) = 0$	0
muy	1	$\log(\frac{2}{1+1}) = \log(1) = 0$	0
grande	1	$\log(\frac{2}{1+1}) = \log(1) = 0$	0

Cuadro 4: Cálculo de IDF para cada palabra del vector 1.

Palabra	**TF**	**IDF**	**TF-IDF**
mobil	$\frac{1}{3}$	0	$\frac{1}{3} \times 0 = 0$
ser	$\frac{1}{3}$	$\log(\frac{2}{3})$	$\frac{1}{3} \times \log(\frac{2}{3}) \approx -1{,}89$
increíble	$\frac{1}{3}$	0	$\frac{1}{3} \times 0 = 0$
pantalla	0	0	$0 \times 0 = 0$
muy	0	0	$0 \times 0 = 0$
grande	0	0	$0 \times 0 = 0$

Cuadro 5: Cálculo de TF-IDF para cada palabra del vector 1.

Vectores Resultantes

Vector resultante del vector 1:

$$[0, -1{,}89, 0, 0, 0, 0]$$

Vector resultante del vector 2:

$$[0, -1{,}89, 0, 0, 0, 0]$$

División de la base de datos

Una vez los datos son convertidos en vectores y están balanceados, la base de datos se divide en dos subconjuntos para entrenar y evaluar el modelo creado:

Conjunto de entrenamiento (80 %): Este subconjunto se utiliza para ajustar los parámetros del modelo. Se proporciona tanto el vector (representación del comentario) como la etiqueta asociada, lo que permite al modelo aprender a relacionar las características del texto con su polaridad correspondiente.

Conjunto de prueba (20 %): Este subconjunto se utiliza para evaluar el desempeño final del modelo. En este caso, solo se proporciona el vector (sin la etiqueta) y el modelo debe predecir la polaridad del texto. La predicción obtenida se compara con la etiqueta original de la base de datos para medir su precisión.

Implementación del modelo

Para la programación de este modelo, se utiliza el algoritmo SVM, ampliamente reconocido por su eficacia en problemas de clasificación binaria. Su capacidad para encontrar un hiperplano óptimo que maximice la separación entre clases lo convierte en una buena elección para este tipo de tareas.

Selección del Kernel

En el caso de problemas de clasificación de texto, como la clasificación de opiniones, se emplea comúnmente un **núcleo lineal**. Esto se debe a que, en general, los textos, cuando se representan como vectores en espacios de alta dimensión, suelen ser separables linealmente. Las representaciones vectoriales de texto, como las obtenidas con la técnica de **TF-IDF**, transforman el texto en vectores que pueden tener miles de dimensiones (una por cada palabra o término en el vocabulario del corpus). En estos espacios de alta dimensión, las clases (en este caso, positivas o negativas) a menudo pueden separarse mediante una línea o un hiperplano.

El kernel lineal tiene varias ventajas en este contexto:

Eficiencia computacional: Al ser un kernel simple, el cálculo y el entrenamiento del modelo SVM con un kernel lineal es más rápido y menos costoso computacionalmente que otros kernels más complejos, como el polinómico o el gaussiano (RBF).

Espacios de alta dimensión: En el espacio vectorial generado por representaciones como TF-IDF, la separación entre clases (por ejemplo, entre tweets positivos y negativos) es más probable que sea lineal, lo que hace que un kernel lineal sea adecuado.

Interpretabilidad: Los modelos SVM con kernel lineal son más fáciles de interpretar. El hiperplano que separa las clases se puede entender directamente como una combinación lineal de las características (palabras) del texto.

Entrenamiento del modelo

Con las características extraídas y los datos representados numéricamente, se puede proceder al **entrenamiento** de un modelo de aprendizaje automático. En esta fase, se elige el **algoritmo** más adecuado para el análisis de sentimientos.

Se ha elegido las Máquinas de Vectores de Soporte como el modelo a estudiar debido a su eficiencia en la clasificación de textos en categorías de sentimientos. Este modelo utiliza un conjunto de datos etiquetado para aprender a identificar patrones en el texto y clasificarlo adecuadamente. Aunque existen otros modelos que también se pueden utilizar en tareas de clasificación de sentimientos, como las Redes Neuronales Profundas, que aprenden de manera jerárquica a representar patrones complejos, o los Árboles de Decisión y Bosques Aleatorios, que también pueden clasificar texto en diferentes categorías, se ha optado por estudiar el rendimiento de las SVM debido a su efectividad y su capacidad para manejar bien los datos de texto en tareas de análisis de sentimientos.

El modelo aprende a partir de un conjunto de datos etiquetados, es decir, datos cuyos sentimientos han sido previamente identificados. Durante el proceso de entrenamiento, el modelo ajusta sus parámetros con el fin de maximizar su capacidad para predecir correctamente los sentimientos de nuevos textos. Este se entrena utilizando el 80 % de

los datos de la base de datos que hemos separado previamente. Para llevar a cabo el entrenamiento, se emplean herramientas de Python, como *scikit-learn*, una biblioteca ampliamente utilizada en la comunidad de aprendizaje automático.

Con el objetivo de optimizar el rendimiento del modelo, se ajustan una serie de parámetros. A continuación, se detallan los parámetros utilizados:

C: Controla el equilibrio entre el ajuste del modelo a los datos de entrenamiento y la maximización del margen de separación entre las clases.

loss: Define la función de pérdida que el modelo utiliza para medir el error durante el entrenamiento.

penalty: Especifica el tipo de regularización que se aplica en el modelo.

max iter: Especifica el número máximo de iteraciones que el algoritmo puede realizar para encontrar la solución óptima.

tol: Define la precisión requerida para considerar que el modelo ha convergido.

El ajuste se ha realizado mediante la técnica de *GridSearchCV*. Esta es utilizada para la optimización de hiperparámetros de un modelo, permitiendo encontrar la combinación de hiperparámetros que da el mejor rendimiento. Por tanto, permite realizar una búsqueda exhaustiva sobre un espacio definido de hiperparámetros y evaluar el rendimiento del modelo con cada combinación. A continuación, se proporcionan las que se han probado:

- `C: [0.1, 1, 10, 100]`
- `loss: [hinge, squared_hinge]`
- `max_iter: [1000, 2000, 5000]`
- `penalty: l2`
- `tol: [0.0001, 0.001, 0.01]`

Y una vez probadas todas las combinaciones posibles, los hiperparámetros que han ofrecido los mejores resultados son:

- `C = 100`
- `loss = hinge`
- `max_iter = 5000`
- `penalty = l2`
- `tol = 0.001`

4.3. Evaluación y resultados

Después de entrenar el modelo, se realiza una **evaluación de su desempeño** utilizando un conjunto de **datos de prueba** (el 20 % de los datos de entrenamiento). Para este propósito, se emplean métricas estándar en análisis de sentimientos que permiten medir la capacidad del modelo para generalizar sobre datos no vistos. A continuación, se describen las métricas clave y el proceso de evaluación:

Métricas de evaluación

Matriz de confusión: Proporciona una representación detallada de las predicciones del modelo, clasificándolas en:

- Verdaderos positivos (TP): Predicciones correctas de la clase positiva.
- Verdaderos negativos (TN): Predicciones correctas de la clase negativa.
- Falsos positivos (FP): Predicciones incorrectas como positivas.
- Falsos negativos (FN): Predicciones incorrectas como negativas.

Esta matriz permite analizar los errores del modelo y ajustar estrategias para mejorarlo.

	Predicción Positiva	Predicción Negativa
Real Positivo	TP	FN
Real Negativo	FP	TN

Cuadro 6: Matriz de confusión

Precisión (*Precision*): Mide la proporción de predicciones correctas para cada clase sobre el total de predicciones realizadas para esa clase.

Sensibilidad (*Recall*): Mide la proporción de verdaderos positivos o negativos sobre el total de casos reales de esa clase.

Puntuación-F1 (*F1-Score*): Es la media armónica entre la precisión y la sensibilidad. Es útil para buscar un equilibrio entre estas dos métricas y evitar que un sesgo hacia una de ellas influya excesivamente.

Precisión global (*Accuracy*): Es el porcentaje de predicciones correctas, considerando todas las clases.

A continuación, para mostrar como se calculan las métricas se proporciona un ejemplo. Supongamos que el modelo genera la siguiente matriz de confusión al evaluar el conjunto de prueba:

	Predicción Positiva	Predicción Negativa
Real Positivo	$50(TP)$	$10(FN)$
Real Negativo	$5(FP)$	$100(TN)$

Cuadro 7: Matriz de confusión obtenida del modelo

A partir de esta matriz, podemos calcular las métricas de evaluación:

- **Precisión global (*Accuracy*):**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{50 + 100}{50 + 100 + 5 + 10} = \frac{150}{165} \approx 0{,}91$$

- **Precisión (*Precision*) para cada clase:**

$$\text{Precision (positive)} = \frac{TP}{TP + FP} = \frac{50}{50 + 5} = \frac{50}{55} \approx 0{,}91$$

$$\text{Precision (negative)} = \frac{TN}{TN + FN} = \frac{100}{100 + 10} = \frac{100}{110} \approx 0{,}91$$

- **Sensibilidad (*Recall*) para cada clase:**

$$\text{Recall (positive)} = \frac{TP}{TP + FN} = \frac{50}{50 + 10} = \frac{50}{60} \approx 0{,}83$$

$$\text{Recall (negative)} = \frac{TN}{TN + FP} = \frac{100}{100 + 5} = \frac{100}{105} \approx 0{,}95$$

- **Puntuación-F1 (*F1-Score*) para cada clase:**

$$\text{F1-Score (positive)} = 2 \cdot \frac{\text{Precision (positive)} \cdot \text{Recall (positive)}}{\text{Precision (positive)} + \text{Recall (positive)}} \approx 0{,}87$$

$$\text{F1-Score (negative)} = 2 \cdot \frac{\text{Precision (negative)} \cdot \text{Recall (negative)}}{\text{Precision (negative)} + \text{Recall (negative)}} \approx 0{,}93$$

Resultados obtenidos del entrenamiento

Una vez entrenado el modelo con el 80 % de los datos, se evalúa su rendimiento utilizando el 20 % restante. Esta **comparación** permite medir la capacidad del modelo para generalizar, utilizando métricas como precisión, sensibilidad y puntuación-F1.

Los resultados se presentan junto con una matriz de confusión, que ofrece una representación visual detallada del rendimiento, destacando tanto los aciertos como los errores del modelo. A continuación, se detallan los resultados obtenidos:

Precisión global: El modelo clasificó correctamente el 93.3 % de todas las muestras.

Sensibilidad y Precisión: De todas las predicciones negativas y positivas, el 93 % se clasificó correctamente.

Puntuación-F1: Con un 93 % en ambos casos, hay un buen equilibrio entre detectar correctamente y no cometer muchos errores.

Como se puede observar, los valores alcanzados para ambas clases están bastante equilibrados, lo que indica que el modelo está funcionando correctamente.

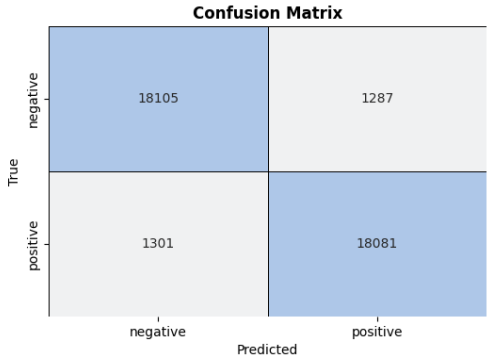

Figura 13: La matriz de confusión del modelo [Fuente propia]

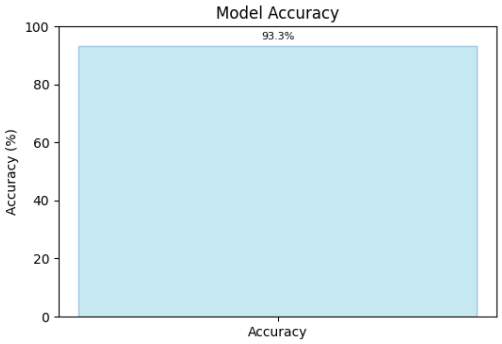

Figura 14: La precisión global del modelo [Fuente propia]

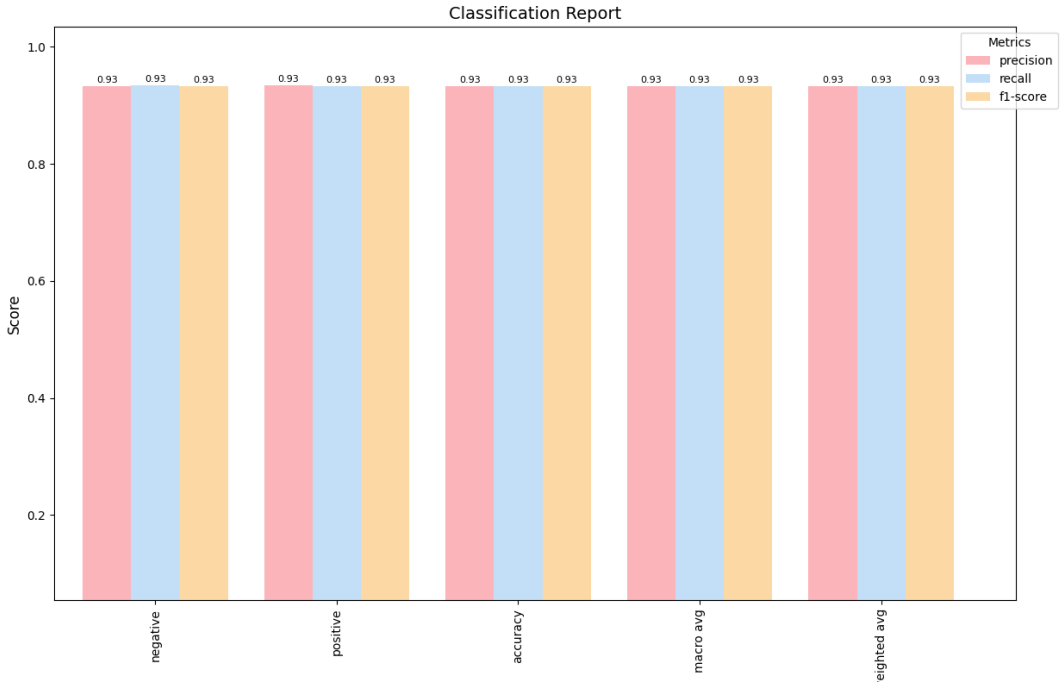

Figura 15: Las métricas restantes del modelo [Fuente propia]

Base de datos a analizar

Tras haber entrenado y evaluado el modelo con una base de datos de opiniones sobre distintos teléfonos de diversas marcas, se procede a evaluar el modelo utilizando otra base de datos etiquetada, pero sin proporcionarle al modelo las etiquetas. Esta ha sido el resultado de la unión de tres bases de datos de reseñas de tres modelos de teléfono, de esta manera se tiene más cantidad de texto a comparar.

Esta base de datos presenta la misma estructura que la utilizada durante la etapa de entrenamiento y evaluación del modelo. Por lo tanto, aplicamos el mismo proceso de transformación de las opiniones en vectores. Finalmente, evaluamos estos vectores mediante el

modelo creado y comparamos sus predicciones con las etiquetas reales obtenidas mediante la puntuación de la reseña para determinar cuántas coinciden, lo que nos permite ver qué tan bien se ajusta el modelo.

Resultados y conclusión

Para realizar la evaluación, se calculan las diferentes métricas utilizadas en la sección anterior a partir de la matriz de confusión. A continuación, se presentan los resultados obtenidos al evaluar el modelo sobre un **conjunto de datos sin modificar para evaluar como el modelo generaliza frente a datos de un origen completamente distinto**.

Los resultados muestran que el modelo presenta un rendimiento notable en el caso binario de clasificación, con una **precisión global** del 87,8 %. Aunque la clase positiva tiene una **precisión**, **sensibilidad** y **puntuación-F1** mucho más altas, el modelo presenta un desempeño significativamente inferior en la clase negativa, con una precisión de solo 36 %. La sensibilidad de la clase negativa es del 74 %, mientras que la de la clase positiva alcanza un 89 %. La puntuación-F1 de la clase negativa se encuentra en un 48 %, en contraste con el 93 % para la clase positiva.

Estos resultados sugieren que el modelo es eficaz para identificar ejemplos de la clase positiva, pero aún **presenta deficiencias en la clasificación de la clase negativa**. Por lo tanto, **se debería realizar ajustes adicionales para optimizar el rendimiento y lograr un mejor equilibrio entre ambas clases**.

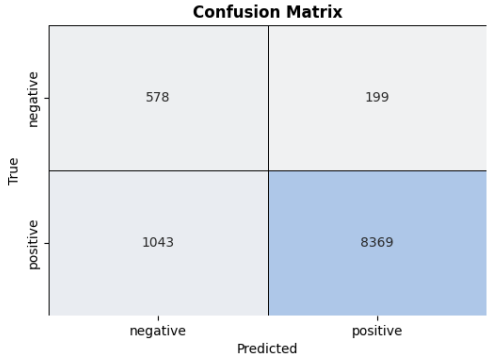

Figura 16: La matriz de confusión del resultado del modelo [Fuente propia]

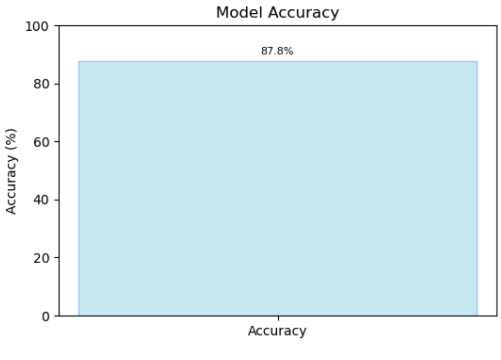

Figura 17: Las métricas restantes del resultado del modelo [Fuente propia]

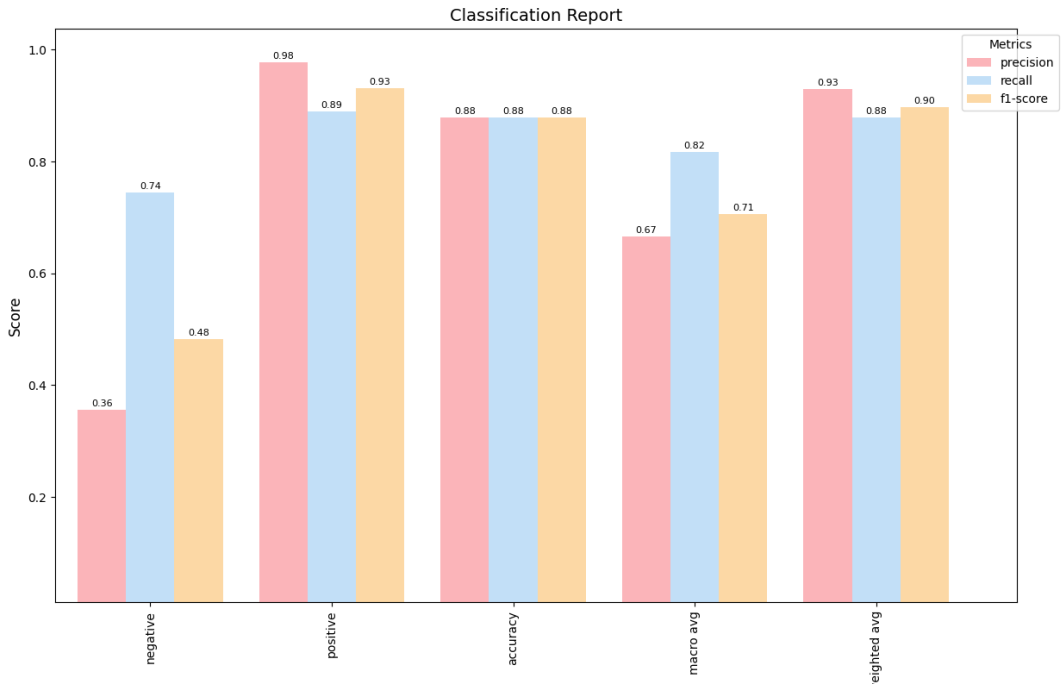

Figura 18: La precisión global del resultado del modelo [Fuente propia]

Propuestas de mejora

Para mejorar la parte práctica, **fortalecería la robustez del modelo** frente a diferentes tipos de conjuntos de datos. El objetivo principal sería optimizar su flexibilidad ante diversas distribuciones de clases, manteniendo su capacidad de generalización y adaptación a distintos escenarios. En este sentido, se proponen las siguientes mejoras:

Mejorar la base de datos con nuevas fuentes: Además de garantizar una adecuada limpieza del texto, es fundamental que los datos recolectados reflejen la diversidad de casos presentes en el problema. Para lograrlo, se propone recopilar múltiples bases de datos provenientes de diversas fuentes, asegurando una representación más amplia y variada de los casos relevantes. Esto permitiría enriquecer el conjunto de entrenamiento y mejorar la capacidad del modelo para generalizar a diferentes escenarios.

Diversificar las estrategias de muestreo y representación de datos: Actualmente se emplea un único tipo de muestreo, el TF-IDF, para representar los datos. Sin embargo, explorar alternativas podría ofrecer ventajas significativas. Esto incluiría probar distintas funciones de transformación más complejas. El objetivo sería maximizar la separación entre clases en el espacio de características y, con ello, mejorar la calidad de las predicciones del modelo.

Incrementar la exploración del espacio de hiperparámetros: Ampliar la búsqueda de configuraciones óptimas para identificar aquellas que maximicen el rendimiento del modelo. Dado que la selección de hiperparámetros desempeña un papel crucial en la calidad de los resultados, sería conveniente emplear métodos más avanzados de búsqueda.

Probar modelos alternativos de clasificación: Evaluar y comparar el desempeño de algoritmos adicionales podría ser una estrategia prometedora. Por ejemplo, modelos

basados en árboles de decisión, como los Bosques Aleatorios (*Random Forest*), son conocidos por su capacidad para manejar datos ruidosos y su eficacia en tareas de clasificación binaria. Comparar estos modelos con el actual en términos de métricas como precisión, sensibilidad y puntuación-F1 podría revelar alternativas con un rendimiento superior para el problema específico.

Estas propuestas buscan abordar tanto las limitaciones actuales como las oportunidades de optimización para mejorar el rendimiento y la capacidad del modelo.

5. Conclusiones

Este trabajo de fin de grado ha permitido una exploración de las Máquinas de Soporte Vectorial, una de las herramientas más poderosas y ampliamente utilizadas en el campo del aprendizaje automático. A lo largo de este proyecto, se ha desentrañado tanto los fundamentos teóricos como las aplicaciones prácticas de las SVM, estableciendo un puente entre la matemática avanzada de la optimización convexa y su uso efectivo en problemas reales, como la clasificación de opiniones en reseñas de teléfonos móviles.

Desde el inicio, se presentó la teoría subyacente a las SVM, explorando su relación con la optimización convexa, que es crucial para entender cómo estos modelos resuelven problemas complejos de clasificación. En el primer capítulo, se discutieron los principios de la optimización, destacando la importancia de conceptos como los problemas primal y dual, las funciones Lagrangianas y los teoremas de dualidad, elementos esenciales para comprender el funcionamiento de las SVM. Estas herramientas matemáticas, como se mostró, no solo son fundamentales para el desarrollo teórico del modelo, sino que también permiten su implementación eficiente en aplicaciones del mundo real.

En el segundo capítulo, el enfoque se centró en los aspectos prácticos de las SVM, detallando cómo estas técnicas superan las limitaciones de otros algoritmos de clasificación, como el perceptrón. Se exploraron los tres casos posibles en la clasificación binaria: datos separables, casi separables y no separables, demostrando la flexibilidad y versatilidad de las SVM en diversos escenarios. Este conocimiento teórico se implementó en un caso práctico real, que consistió en la clasificación de opiniones de reseñas sobre teléfonos móviles, mostrando cómo las SVM pueden ser aplicadas eficazmente a problemas de análisis de sentimientos, donde los datos se presentan de manera compleja y ruidosa.

Es importante recalcar que **la elección de las SVM para el modelo de este problema no se debe a que sea el método más adecuado o el mejor para realizar el análisis, sino simplemente porque se buscaba una aplicación práctica que permitiera ilustrar y hacer más visual la parte teórica del trabajo**. El objetivo principal fue crear una demostración clara y comprensible de cómo funcionan las SVM en la práctica, no necesariamente a encontrar la solución más óptima para la clasificación de opiniones. De esta manera, se facilitó la comprensión del modelo y su implementación, sirviendo como una plataforma para explorar sus capacidades en un escenario real.

En cuanto a los resultados obtenidos, el modelo mostró un rendimiento destacado, con una precisión global del 87,8 %. Sin embargo, se observó que la clasificación de la clase negativa presentaba deficiencias significativas, con una precisión mucho más baja (36 %) en comparación con la clase positiva, que alcanzó un rendimiento mucho más alto (93 % en puntuación-F1). Este hallazgo indica que, aunque el modelo es eficiente para identificar opiniones positivas, aún necesita ajustes significativos para equilibrar la clasificación de ambas clases de manera más justa y precisa.

Las propuestas de mejora se centran en diversos aspectos clave para fortalecer el modelo. La primera recomendación fue mejorar la base de datos, asegurando su diversidad y representatividad para aumentar la capacidad del modelo de generalizar a distintos tipos de datos. En segundo lugar, se sugirió explorar distintas técnicas de muestreo y representación de datos, como funciones de transformación más complejas que podrían mejorar la separación entre clases y, por ende, la calidad de las predicciones. Además, se destacó la importancia de expandir la exploración del espacio de hiperparámetros, utilizando métodos más avanzados para optimizar la configuración del modelo. Finalmente, se propuso

probar otros algoritmos de clasificación, como los Bosques Aleatorios, para comparar su rendimiento con el de las SVM y determinar si podrían superar algunas de las limitaciones observadas.

En conclusión, este trabajo no solo ha logrado mostrar una aplicación de las SVM en la clasificación de datos complejos, sino también ha proporcionado una comprensión de los principios matemáticos que sustentan esta técnica. La combinación de teoría y práctica a través de la aplicación a un caso real ha permitido identificar áreas de mejora y nuevas oportunidades de optimización. A lo largo del proyecto, se ha dado un equilibrio entre la comprensión de los conceptos fundamentales y su implementación práctica, lo cual podria consituir una base para trabajos futuros. A medida que se sigan implementando las propuestas de mejora, se esperaria que el modelo fuera capaz de ofrecer una clasificación más equilibrada y precisa, adaptándose mejor a las demandas de análisis de sentimientos y otros problemas de clasificación en el mundo real.

Bibliografía

[1] Marsland, S.: *Machine Learning: An Algorithmic Perspective, Second Edition* (2a ed.). Chapman & Hall/CRC, 2014.

[2] Steinwart, I., & Christmann, A.: *Support Vector Machines* (2008a ed.). Springer, 2014.

[3] Soria, Sánchez-Montñés, Gamero, Castillo, Cano, N. (2023). *Sistemas de aprendizaje automático*. RA-MA.

[4] Kowalczyk, A. (2017). *Support Vector Machines Succinctly*. Release date: October 24, 2017.

[5] Básicos, 1. 1. Conceptos: *Fundamentos de Optimización*. Recuperado el 30 de diciembre de 2024, de `https://www.dmae.upct.es/~paredes/am_ti/apuntes/guia_foe.pdf`

[6] Rodriguez, A. A.: *Optimización convexa*. Recuperado el 30 de diciembre de 2024, de `https://biblus.us.es/bibing/proyectos/abreproy/5442/fichero/memoria.pdf`

[7] Universidad de Barcelona. *Optimización Económica*. Recuperado el 1 de enero de 2025, de `http://www.ub.edu/matheopt/optimizacion-economica/`

[8] Universidad de Barcelona. *Optimización con restricciones de igualdad*. Recuperado el 1 de enero de 2025, de `http://www.ub.edu/matheopt/optimizacion-economica/optimizacion-con-restricciones-de-igualdad`

[9] Universidad de Barcelona. *Condición de Karush-Kuhn-Tucker*. Recuperado el 1 de enero de 2025, de `http://www.ub.edu/matheopt/optimizacion-economica/condicion-de-khun-tucker`

[10] Paredes, P. L. [@ProfesorLuisParedes]. *El teorema de Heine-Borel*. Youtube. Recuperado el 1 de enero de 2025, de `https://www.youtube.com/watch?v=mxNWb32UHes`

[11] MathPures [@MathPuresChannel]. *Si f es Continua en un Compacto Alcanza su Máximo y Mínimo Curso de Análisis Matemático*. Youtube. Recuperado el 1 de enero de 2025, de `https://www.youtube.com/watch?v=IRhT7kavrGs`

[12] Universitat Politècnica de València-UPV. *Algoritmo perceptrón-UPV*. Youtube. Recuperado el 3 de enero de 2025, de
https://www.youtube.com/watch?v=8dXYLwPiqME.

[13] Weinberger, K. *Lecture 6 "perceptron convergence proof"-Cornell CS4780 SP17*. Youtube. Recuperado el 3 de enero de 2025, de
https://www.youtube.com/watch?v=kObhWlqIeD8.

[14] Amat Rodrigo, J. (2025). *Máquinas de Vector Soporte (SVM) con Python*. Disponible en
https://www.cienciadedatos.net/documentos/py24-svm-python.html

[15] Wikipedia contributors. *Problema de optimización*. Wikipedia, The Free Encyclopedia. Recuperado el 1 de enero de 2025, de
https://es.wikipedia.org/wiki/Problema_de_optimización

[16] Wikipedia contributors. *Método kernel*. Wikipedia, The Free Encyclopedia. Recuperado el 1 de enero de 2025, de
https://es.wikipedia.org/wiki/Método_Kernel

[17] Wikipedia contributors. *Espacio de Hilbert*. Wikipedia, The Free Encyclopedia. Recuperado el 1 de enero de 2025, de
https://es.wikipedia.org/wiki/Espacio_de_Hilbert

[18] Wikipedia contributors. *Perceptrón*. Wikipedia, The Free Encyclopedia. Recuperado el 1 de enero de 2025, de
https://es.wikipedia.org/wiki/Perceptrón

[19] Wikipedia contributors. *Máquina de vectores de soporte*. Wikipedia, The Free Encyclopedia. Recuperado el 2 de enero de 2025, de
https://es.wikipedia.org/wiki/Máquina_de_vectores_de_soporte

[20] Viñas, U.: Repositorio código. Github.io. Recuperado el 3 de enero de 2025, de
https://github.com/udanevinas/svm_tfg

[21] Kaggle.com: Amazon reviews unlocked mobile phones. Recuperado el 4 de enero de 2025, de
https://www.kaggle.com/datasets/PromptCloudHQ/amazon-reviews-unlocked-mobile-phones

[22] Kaggle.com: Apple iPhone SE reviews and ratings. Recuperado el 4 de enero de 2025, de
https://www.kaggle.com/datasets/kmldas/apple-iphone-se-reviews-ratings

[23] Kaggle.com: iPhone 14 128GB Flipkart user reviews. Recuperado el 4 de enero de 2025, de
https://www.kaggle.com/datasets/nuhmanpk/iphone-14-128gb-flipkart-user-reviews

[24] Kaggle.com: iPhone 16 review with customer rating. Recuperado el 4 de enero de 2025, de
https://www.kaggle.com/datasets/nuhmanpk/iphone-16-review-with-customer-rating