FACULTAD DE MATEMÁTICAS

DEPARTAMENTO: CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

Trabajo Fin de Grado:

Fundamentos Matemáticos de los Métodos Kernel para Aprendizaje Supervisado

Autor:
Ana López Díaz

Dirigido por:
Miguel Ángel Gutiérrez Naranjo

2017 - 2018

Resumen

El objetivo del presente estudio es resaltar la importancia de las máquinas de vector soporte, conjunto de técnicas de aprendizaje supervisado aplicadas a resolver tareas de clasificación. Además, se enfatizan algunos de los enlaces entre los métodos de optimización matemática y la clasificación supervisada. Muchas áreas diferentes de optimización matemática desempeñan un papel central en los métodos de clasificación supervisada. Por otra parte, la optimización matemática resulta extremadamente útil para abordar cuestiones importantes en la clasificación, como la identificación de variables relevantes o la mejora de la interpretabilidad de los clasificadores.

Antes que nada, se hace un desarrollo teórico de los fundamentos matemáticos de las máquinas de vector soporte [1]. Se comienza el desarrollo por el caso más sencillo, es decir, el caso de clasificación binaria y, atendiendo al tipo de separabilidad de los ejemplos de entrada, se consideran distintas opciones. Así, en primer lugar, se aborda el caso ideal de ejemplos linealmente separables para, seguidamente, abordar el caso de ejemplos no linealmente separables, donde las SVM demuestran su gran potencialidad. Para este último caso, se introducen las funciones kernel que hace que las SMV sean aplicables para cualquier conjunto de datos.

Finalmente, estas técnicas estudiadas son utilizadas para la clasificación de mamografías, lo cual permite construir un clasificador que separe la clase de tumores benignos de la clase de tumores malignos, a partir de un conjunto de datos tomados de "UCI Machine Learning Repository: Breast Cancer Wisconsin". De esta forma, el objetivo será construir el mejor clasificador para dicho conjunto de datos.

[1] También conocido por las siglas SVM.

Abstract

The aim of the present study is to highlight the importance of support vector machines, which are a set of supervised learning techniques applied to solve classification tasks. In addition, some links between mathematical optimization methods and supervised classification are emphasized. Many different areas of mathematical optimization play a central role in the methods of supervised classification. On the other hand, the mathematical optimization turns out to be extremely useful in order to deal with important issues in classifications, as the identification of relevant variables or the improvement of the interpretability of classifiers.

First of all, a theoretical development of the mathematical foundations of support vector machines (SVM) is presented. The development begins for the simplest case, that is to say, the case of binary classification and attending the type of separability of the input examples, we can consider different options. Firstly, the ideal case of linearly separable examples are approached, afterwards, the case of not linearly separable examples is approached as well, where the SVM demonstrates his great potential. For the latter case, to apply the SVM for any set of information, kernel functions are introduced.

Finally, these techniques are used for the classification of mammograms, which allows to construct a classifier that separates the class of benign tumours from the class of malignant tumours, from a set of data taken of " UCI Machine Learning Repository: Breast Cancer Wisconsin". In this way, the aim will be to construct the best classifier for the set of data mentioned before.

Índice general

Capítulo 1

Introducción

La construcción de máquinas capaces de aprender de la experiencia ha sido durante mucho tiempo objetivo de debate tanto filosófico como técnico, ya que la posibilidad de crear máquinas que adquieran conocimiento, que fueran más allá del conocimiento del diseñador del algoritmo, ha fascinado desde siempre.

Desde el punto de vista filosófico, el problema de aprender de los datos ha sido investigado por los filósofos a lo largo de la historia, bajo el nombre de inferencia inductiva. Aunque fue en el siglo XX, gracias al trabajo fundamental de Karl Popper, cuando se reconoció la inducción pura como imposible, a no ser que se asumiese algún conocimiento previo. Además, hay una larga historia del estudio de este problema dentro del marco estadístico y del área de la inteligencia artificial. Alan Turing [1] planteó la idea de que las máquinas pueden pensar en 1950, pero fue pocos años más tarde cuando comenzaron a desarrollarse los primeros ejemplos de estas máquinas, como el borrador de Arthur Samuel [2],que fue un ejemplo temprano de aprendizaje de refuerzo, o el perceptrón de Frank Rosenblatt [3]. Hasta que finalmente, las Support Vector Machine [1]) fueron presentadas por Vladimir Vapnik y sus colaboradores en el COLT [4], a principio de los años noventa.

Desde el punto de vista técnico, las máquinas tienen un nivel significativo de capacidad de aprendizaje, aunque los límites de esta capacidad no están claramente definidos. Asimismo, la disponibilidad de sistemas de aprendizaje es de gran importancia, ya que hay muchas tareas que no pueden ser resueltas mediante técnicas de programación clásica al no haber un modelo matemático del problema disponible.

[1]En español se le denomina Máquinas de Vector Soporte. A lo lago del trabajo, se va a denotar como SVM

Una estrategia alternativa para resolver este tipo de problemas es que la máquina intente aprender la funcionalidad de entrada/salida a partir de ejemplos.

1.1. Aprendizaje Automático

Como se mencionó anteriormente, desde que se inventaron las máquinas, nos hemos preguntado si podrían hacerse para aprender. Si se podría entender cómo programarlas para que aprendan (para mejorar automáticamente con la experiencia), por ejemplo, para saber qué tratamientos son más efectivos para nuevas enfermedades en función de los registros médicos. Además, una comprensión exitosa de cómo hacer que las máquinas aprendan abriría múltiples usos nuevos de estas.

El Machine Learning (*aprendizaje automático*) busca resolver la cuestión de cómo construir programas de máquinas que mejoren automáticamente con la experiencia. En los últimos años, se han desarrollado muchas aplicaciones exitosas de aprendizaje automático, desde programas de extracción de datos que detectan transacciones fraudulentas con tarjeta de crédito, sistemas de filtrado de información que aprenden las preferencias de lectura de los usuarios, hasta vehículos autónomos que aprenden a conducir en autopistas públicas. Al mismo tiempo, han habido avances importantes en la teoría y los algoritmos que forman los cimientos de este campo.

Se basa en conceptos y resultados de muchos campos, incluidas la estadística, la inteligencia artificial, la filosofía, la teoría de la información, la biología, la ciencia cognitiva, la complejidad computacional y la teoría del control. La teoría del aprendizaje automático pretende responder al siguiente tipo de preguntas: "¿Cómo varía el rendimiento del aprendizaje con la cantidad de ejemplos de entrenamiento presentados?", "¿Qué algoritmos de aprendizaje son los más apropiados para varios tipos de tareas de aprendizaje?".

A día de hoy, no se sabe cómo hacer que las máquinas aprendan tan bien como la gente aprende. Sin embargo, se han inventado algoritmos que son efectivos para ciertos tipos de tareas de aprendizaje y está empezando a surgir una com-

prensión teórica del aprendizaje. También han comenzado a aparecer aplicaciones comerciales significativas para problemas tales como el reconocimiento de voz. A continuación, se va a definir qué se entiende por aprendizaje de manera amplia, para incluir cualquier programa que mejore su ejecución en alguna tarea a través de la experiencia,

Definición 1. *Se dice que un programa aprende de la experiencia E con respecto a alguna clase de tareas T y la medida de rendimiento P, si su ejecución de tareas en T, medido por P, mejora con la experiencia E.*

En general, para tener un problema de aprendizaje bien definido, debemos identificar las tres características siguientes: la clase de tareas, la medida del rendimiento que se debe mejorar y la fuente de la experiencia. Diseñar un enfoque de aprendizaje automático implica una serie de opciones de diseño, que incluyen elegir el tipo de experiencia de entrenamiento, la función de destino que se debe aprender, una representación para esta función objetivo y un algoritmo para aprender la función objetivo a partir de ejemplos de entrenamiento. Es más, implica una búsqueda a través de un espacio de posibles hipótesis para encontrar la hipótesis que mejor se ajusta a los ejemplos de capacitación disponibles y otras limitaciones o conocimientos previos.

Los algoritmos de aprendizaje automático han demostrado ser de gran valor práctico en una variedad de dominios de aplicaciones. Son especialmente útiles en:

1. Problemas de Data Mining (*minería de datos*)[2], donde grandes bases de datos pueden contener valiosas regularidades implícitas que se pueden descubrir automáticamente. Por ejemplo, analizar resultados de tratamientos médicos de bases de datos de pacientes.

2. Dominios en los que los humanos podrían no tener el conocimiento necesario para desarrollar algoritmos efectivos. Por ejemplo, reconocimiento de rostros humanos a partir de imágenes.

3. Dominios donde el programa debe adaptarse dinámicamente a las condiciones cambiantes. Por ejemplo,adaptarse a los cambiantes intereses de lectura de las personas.

[2]Una introducción de Data Mining la podemos encontrar en [5]

El aprendizaje automático se divide en dos áreas principales: aprendizaje supervisado y aprendizaje no supervisado, estos conceptos hacen referencia a qué queremos hacer con los datos. Uno de los usos más extendidos del aprendizaje supervisado consiste en hacer predicciones a futuro basadas en comportamientos o características que se han visto en los datos ya almacenados (el histórico de datos). El aprendizaje supervisado permite buscar patrones en datos históricos relacionando todos los campos con un campo especial, llamado campo objetivo. En cuanto al aprendizaje no supervisado, usa datos históricos que no están etiquetados. El fin es explorarlos para encontrar alguna estructura o forma de organizarlos.

A lo largo de este trabajo, se va a estudiar un caso particular dentro del aprendizaje automático supervisado, que son las SVM. En este contexto, se puede diferenciar distintos problemas en función del tipo de salida que generen, de manera que podemos establecer la siguiente clasificación.

Definición 2. *Un problema de aprendizaje con salidas binarias se define como problema de clasificación binaria, uno con un número finito de categorías como clasificación de clases múltiples y cuando estas salidas son valores reales se conoce como un problema de regresión.*

Así, un problema de clasificación que es una técnica muy útil, usada en diversos campos como el de reconocimiento de patrones, predice una categoría, mientras que uno de regresión predice un número. Las SVM se pueden aplicar tanto para problemas de casificación como de regresión, no obstante, sólo se va a hacer un estudio de las SVM aplicadas a problemas de clasificación.

Como se mencionó anteriormente, la clasificación supervisada cuenta con un conocimiento a priori, es decir, la tarea de clasificar un objeto dentro de una categoría o clase cuenta con modelos ya clasificados. Por lo tanto, se pueden diferenciar dos fases dentro de este tipo de clasificación: en primer lugar, se parte de un conjunto de entrenamiento con el que se diseña el clasificador, que va a ser el modelo o regla para la clasificación, y en segundo lugar, se clasifican los objetos de los que se desconoce la clase de pertenencia (el conjunto de dichos objetos se conoce como conjunto test).

En definitiva, en la clasificación supervisada ([5],[6]), dado un conjunto de objetos en Ω divididos en un conjunto de clases C con el objetivo de, a partir de ellos,

clasificar nuevos objetos. Cada objeto $i \in \Omega$ tiene asociado un par (x_i, y_i), donde x_i es el vector predictor, que toma valores en un conjunto X e $y_i \in \Omega$ va a ser la clase a la que pertenece el objeto i . Generalmente, suponemos que $X \subseteq \mathbb{R}^p$. De aquí en adelante, se usa el término variable para referirse a cada componente del vector predictor. Sin embargo, no toda a información sobre los objetos en Ω está disponible, sólo se conoce la clase a la que pertenecen los objetos i que están en un subconjunto $I \subset \Omega$, llamado conjunto de entrenamiento. Con esta información, se busca una regla de clasificación, en definitiva, una función $y : X \to C$, que asigne la etiqueta $y(x) \in C$ al vector predictor x, $\forall x$.

En su forma básica, C consiste en un conjunto finito de valores nominales, sin una clasificación intrínseca (por ejemplo, C={benigno, maligno}), aunque parte de la teoría se extiende al caso en que C es un conjunto finito equipado con una relación de orden, o un segmento de la línea real. En este último caso, tendríamos un problema de regresión en su lugar [7], como ya se adelantó.

La clasificación supervisada se ha aplicado con éxito en muchos campos diferentes. Se encuentran ejemplos en la categorización de texto [8], la indexación de documentos, la clasificación de páginas web y el filtrado de correo no deseado. También se encuentran muchos ejemplos en el ámbito de la biología y la medicina, como la clasificación de datos de expresión génica [9, 10], detección de homología [11], predicción de interacción proteína-proteína [12, 13], cerebro anormal clasificación de actividad [14] y diagnóstico de cáncer [15, 16]; la visión artificial [17, 18]; la agricultura [19]; o la química [20], por citar algunos campos y referencias.

Las aplicaciones empresariales han tenido un comienzo posterior, a pesar de esto, hoy en día podemos encontrar muchas aplicaciones de clasificación supervisada en marketing, banca, entre otros [21, 22, 23, 24]. Las aplicaciones empresariales típicas son la puntación de crédito [25], la quiebra [26], detección de fraude [27], orientación de clientes [28], lealtad del cliente [29, 30], análisis de cesta de mercado [31], recomendador sistemas [32], gestión de ingresos [33], cancelación de reservas de servicios [34], calificaciones de riesgo país [35], predicción de la salud costos [36] o previsión del mercado de valores [37].

Cabe destacar la importancia de la optimización matemática ([38, 39]), ya que va a ser el núcleo de estos métodos. El éxito de la optimización matemática cuando se aplica a la clasificación supervisada tiene uno de sus principales exponentes

en SVM ([40], [41], [42], [43]), una técnica arraigada en la teoría del aprendizaje estadístico ([42], [43]) que ha demostrado ser uno de los métodos más modernos para el aprendizaje supervisado, pero no va a ser el único. Para el caso de dos clases, SVM tiene como objetivo separar ambas clases por medio de un hiperplano que maximiza el margen, es decir, el ancho del banda que separa los dos conjuntos.

1.1.1. Algunos métodos

Dentro del ámbito del Aprendizaje Automático y más concretamente de la clasificación supervisada, se han desarrollado numerosos métodos entre los cuales podemos destacar los siguientes:

Árboles de decisión

El aprendizaje del árbol de decisión es uno de los métodos más utilizados y prácticos para la inferencia inductiva. Es un método para aproximar las funciones objetivo de valores discretos, en el que la función aprendida se representa mediante un árbol de decisiones, que es resistente para datos dispares y capaz de aprender expresiones disyuntivas. Las ramas constituyen los patrones reconocidos en el proceso de aprendizaje, mientras que en las hojas de las ramas se sitúan las predicciones para cada patrón.

Este método buscan un espacio de hipótesis completamente expresivo para evitar las dificultades de los espacios de hipótesis restringidos. Su sesgo inductivo es una preferencia por árboles pequeños sobre árboles grandes.

Los árboles de decisión se han aplicado con éxito a una amplia gama de tareas, desde aprender a diagnosticar casos médicos hasta aprender a evaluar el riesgo de crédito de los solicitantes de préstamos.

Un árbol de decisión está formado por un conjunto de nodos de decisión (interiores) y de nodos-respuesta (hojas), donde:

- Un nodo de decisión está asociado a uno de los atributos y tiene 2 o más ramas que salen de él, cada una de ellas representando los posibles valores

que puede tomar el atributo asociado. De alguna forma, un nodo de decisión es como una pregunta que se le hace al ejemplo analizado, y dependiendo de la respuesta que de, una u otra de las ramas salientes.

- Un nodo-respuesta está asociado a la clasificación que se quiere proporcionar, y nos devuelve la decisión del árbol con respecto al ejemplo de entrada.

Sin embargo, la construcción del árbol de decisión no es única, si aplicamos una estrategia u otra a la hora para decidir el orden de las preguntas sobre los atributos podemos encontrar árboles muy dispares. De manera que, entre todos los posibles árboles, se busca aquellos que cumplan las mejores características. Un método para construir árboles de decisión que presentan muy buenas características (buen balanceo y tamaño pequeño) es el algoritmo ID3. Este último, construye un árbol de decisión de arriba a abajo, de forma directa y basándose en los ejemplos iniciales proporcionados, usando el concepto de ganancia de información para seleccionar el atributo más útil en cada paso.

Redes Neuronales Artificiales

Una Red Neuronal Artificial ([44]) es un modelo matemático inspirado en el comportamiento biológico de las neuronas y en cómo se organizan formando la estructura del cerebro. La unidad fundamental de la red neuronal es el perceptrón, que es un elemento que tiene varias entradas con un cierto peso. Respecto a su funcionamiento, el cerebro puede ser visto como un sistema inteligente que lleva a cabo tareas de manera distinta a como lo hacen las máquinas actuales. Éstas últimas son muy rápidas en el procesamiento de la información, pero existen tareas muy complejas, como el reconocimiento y clasificación de patrones, que demandan demasiado tiempo y esfuerzo en las máquinas más potentes de la actualidad. Sin embargo, el cerebro humano es más eficiente para resolverlas, muchas veces sin aparente esfuerzo (por ejemplo, el reconocimiento de un rostro familiar entre una multitud de otros rostros).

Las redes neuronales artificiales (ANN[3]) proporcionan un método general y práctico para el aprendizaje de ejemplos de valores reales, valores discretos y valo-

[3]Estas siglas están tomadas del nombre inglés, $Artificial Neural Network$, aunque también se pueden encontrar las siglas en español RNA.

res de vectores, que es resistente a errores en los datos de entrenamiento. Se aplica con éxito a problemas tales como la interpretación de escenas visuales, el reconocimiento de voz y el aprendizaje de estrategias de control de robots. Para ciertos tipos de problemas, como aprender a interpretar datos complejos de sensores del mundo real, las redes neuronales artificiales se encuentran entre los métodos de aprendizaje más efectivos actualmente conocidos.

El estudio de las redes neuronales artificiales se inspiró en los sistemas de aprendizaje biológico, que están formados por redes muy complejas de neuronas interconectadas. Por ello, se construyen a partir de un conjunto de unidades simples (llamados nodos o neuronas) densamente interconectadas que están organizados en capa, donde cada unidad toma una cantidad de entradas de valores reales y produce una sola salida de valor real. Cada neurona está conectada con otras neuronas mediante enlaces de comunicación, cada uno de los cuales tiene asociado un peso. El algoritmo "Propagación hacia atrás" es el método de aprendizaje en red más común.

Las ANN son sistemas adaptativos que aprenden de la experiencia, esto es, aprenden a llevar a cabo tareas mediante un entrenamiento o aprendizaje. Gracias a este entrenamiento, crean su propia representación interna del problema (por ello se dice que son autoorganizadas), con el fin de generar estos casos a otros nuevos.

Además, las ANN realizan el procesamiento de la información en paralelo, lo que permite que muchas neuronas pueden estar funcionando al mismo tiempo. Es ahí donde reside una parte fundamental de su poder de procesamiento. El tipo de representación de la información que manejan, tanto en los pesos de las conexiones como en las entradas y salidas de información, es numérica. En definitiva, las ANN se inspiran en la estructura del sistema nervioso, con la intención de construir sistemas de procesamiento de la información paralelos, distribuidos y adaptativos que pueden presentar un cierto comportamiento inteligente.

Aprendizaje Bayesiano

El razonamiento bayesiano proporciona un enfoque probabilístico para la inferencia. Se basa en la suposición de que las cantidades de interés, se rigen por distribuciones de probabilidad y que las decisiones óptimas se pueden tomar ra-

zonando sobre estas probabilidades junto con los datos observados. Consiste en sopesar las diferentes hipótesis y asignarles una probabilidad de acuerdo a los datos de entrenamiento que clasifica correctamente. Por esa razón, forma la base para aprender algoritmos que manipulan directamente las probabilidades, así como un marco para analizar el funcionamiento de otros algoritmos que no manipulan explícitamente las probabilidades.

Los algoritmos de aprendizaje bayesiano que calculan probabilidades explícitas para hipótesis, como el clasificador ingenuo de Bayes, se encuentran entre los enfoques más prácticos para ciertos tipos de problemas de aprendizaje.

En el aprendizaje automático a menudo nos interesa determinar la mejor hipótesis desde algún espacio H, dados los datos de entrenamiento observados D. El teorema de Bayes va a proporcionar una forma de calcular la probabilidad de una hipótesis en función de su probabilidad previa, las probabilidades de observar varios datos dada la hipótesis y los datos observados; de forma que ésta va a ser hipótesis óptima en el sentido de que ninguna otra hipótesis es más probable.

En el aprendizaje bayesiano, cada ejemplo de entrenamieto observado puede incrementar o disminuir la probabilidad estimada de una hipótesis. El conocimiento a priori se obtiene a partir de la probabilidad para cada hipótesis candidata y la distribución de la probabilidad sobre los datos observados para cada posible hipótesis. Además, el conocimiento a priori puede ser combinado con los datos observados para determinar la probabilidad final de una hipótesis.

Una dificultad práctica en la aplicación de métodos bayesianos es que, generalmente, requieren un conocimiento inicial de muchas probabilidades. Cuando estas probabilidades no se conocen de antemano, a menudo se estiman basándose en el conocimiento previo, datos disponibles previamente y suposiciones sobre la forma de las distribuciones subyacentes. Una segunda dificultad práctica es el significativo coste computacional requerido para determinar la hipótesis óptima de Bayes en el caso general.

En particular, una SVM es un algoritmo de aprendizaje automático capaz de identificar un separador. Es decir, si consideramos un conjunto finito de vectores de $\mathbb{R}^n$ separados en dos clases, construir un clasificador consiste en construir una función que toma como valores de entrada un vector del conjunto y devuelve como

salida la pertenencia del vector a una de las clases. Las SVM proporcionan una solución a este problema con buenas propiedades de generalización.

SVM tiene algunas ventajas en comparación con otros métodos. Primero, resuelve un problema de programación cuadrática (QP), que asegura que una vez que se obtiene su solución, es la solución única (global). En segundo lugar, la escasez de SVM asegura una mejor generalización. En tercer lugar, en lugar del principio empírico de minimización de riesgos, SVM implementa el principio de minimización del riesgo estructural que minimiza el límite superior del error de generalización. En cuarto lugar, tiene una clara intuición geométrica en la tarea de clasificación.

Por consiguiente, podemos resaltar la relación existente entre SVM y perceptrón, o neurona artificial, cuya diferencia es que el perceptrón busca reducir el número de errores en cada iteración.

La primera idea del diseño de SVM fue partir de un tipo de funciones sencillas buscando una solución óptima y, posteriormente, ampliar el tipo de funciones que pueden aprenderse usando kernel sin aumentar excesivamente la complejidad del proceso. En particular, las SVM buscan el hiperplano con menos riesgo desde el punto de vista estructural, de manera que el objetivo va a ser maximizar el margen de la solución; esto lo podemos entender como buscar un clasificador de forma que separe en mayor medida las dos clases. Dicho proceso de maximizar nos va a llevar a un problema de optimización.

Capítulo 2

SVM y Métodos Kernel

2.1. Máquinas Vectoriales de Soporte

SVM es un algoritmo de aprendizaje automático capaz de identificar un clasificador. En el aprendizaje supervisado, la máquina de aprendizaje recibe un conjunto de ejemplos de entrenamiento (o entradas) con etiquetas asociadas (o valores de salida). Por lo general, los ejemplos están en forma de vectores característicos, de modo que el espacio de entrada es un subconjunto de $\mathbb{R}^n$. Una vez que los vectores característicos están disponibles, se podría elegir un número de conjuntos de hipótesis para el problema. Entre estas, las funciones lineales son las mejor entendidas y las más simples de aplicar. Por ello nos referiremos a las máquinas de aprendizaje que utilizan hipótesis que forman combinaciones lineales de las variables de entrada como máquinas de aprendizaje lineal, ya que en estos casos las máquinas pueden representarse en una forma particularmente útil que denominaremos representación dual.

Como se citó anteriormente, SVM fueron introducidas por Vapnik a partir de sus trabajos sobre las teorías del aprendizaje estadístico, en los que acotaba el error en función de la complejidad del espacio de hipótesis. Sin embargo, es su capacidad para producir buenos modelos, para múltiples tipos de aplicaciones prácticas, la que le ha llevado a tener una gran popularidad. En un principio, se diseñaron exclusivamente para resolver problemas de clasificación binaria, pero con el tiempo su uso se extendió a tareas de regresión, clasificación multi-categóricas, agrupamiento, etc. SVM busca producir predicciones en las que se pueda tener mucha confianza a costa de producir ciertos errores, es decir, pretende construir modelos

confiables lo que se conoce como el principio Minimización del Riesgo Estructural (un modelo que estructuralmente tenga poco riesgo de cometer errores ante datos futuros). En definitiva, las SVM se utilizan para predecir datos una vez que se haya entrenado la máquina, de esta forma, su resolución se basa en una primera fase de entrenamiento (se les informa con muchos ejemplos) y una segunda fase de uso para la resolución de problemas.

2.1.1. Clasificador de margen máximo

Para comenzar, se definen algunos conceptos para referirnos a las entradas, salidas, conjuntos de entrenamiento, etc. A las cantidades introducidas para describir los datos las llamaremos características, mientras que las cantidades originales serán los atributos.

Definición 3. *Usualmente X se conoce como el espacio de entrada, Y para el dominio de salida y F como el espacio de características. Generalmente, tendremos $X \subseteq \mathbb{R}^n$, mientras que para la clasificación binaria $Y = \{-1, 1\}$, para la clasificación en m clases $Y = \{1, 2, ..., m\}$, y para la regresión $Y \subseteq \mathbb{R}$. Un conjunto de entrenamiento es una colección de ejemplos de entrenamiento, que también se llaman datos de entrenamiento. Se suele denotar por*

$$S = \{(x_1, y_1), ..., (x_l, y_l)\} \subseteq (X \times Y)^l,$$

donde l es la cantidad de ejemplos. Nos referimos a las x_i como ejemplos o instancias y a las y_i como sus etiquetas. El conjunto de entrenamiento S es trivial si las etiquetas de todos los ejemplos son iguales. Hay que tener en cuenta que si X es un espacio vectorial, los vectores de entrada son vectores de columnas al igual que los vectores de ponderación. Si deseamos formar un vector fila a partir de x_i podemos tomar la traspuesta x_i'.

Se va a comenzar el estudio analizando el caso más sencillo en el que se tienen dos clases linealmente separables. El procedimiento general a seguir va a ser construir un hiperplano que separe el conjunto de datos en dos semiespacios lo más amplios posibles. Para ello, se va a definir el concepto de hiperplano (que va a ser una herramienta fundamental en el estudio).

Definición 4. *En un espacio p-dimensional, un hiperplano es una variedad afín plana (p-1)-dimensional. Cuya ecuación general puede ser expresada como:*

$$w_1x_1 + w_2x_2 + ... + w_px_p + b = 0$$

Se parte de un conjunto de muestras dado

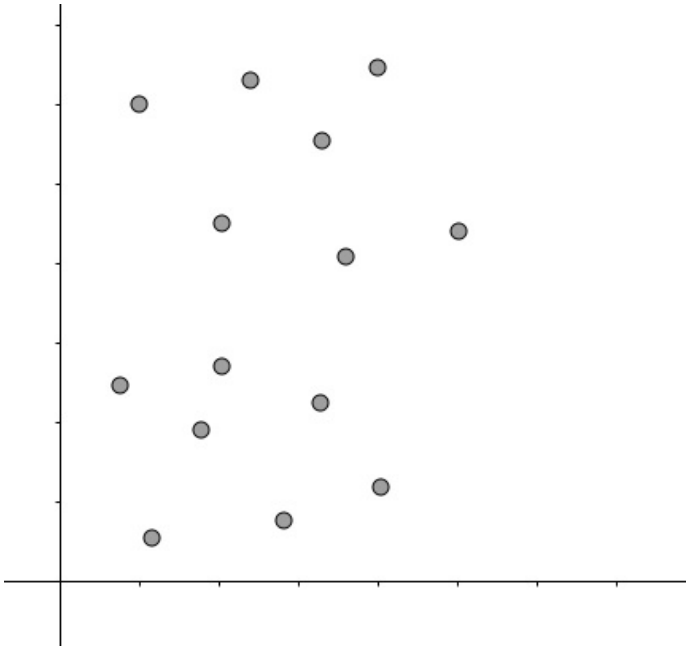


Figura 2.1: Representa un conjunto de muestra cualquiera.

que están etiquetadas en dos clases distintas, y dado un nuevo individuo se busca saber a qué clases de las anteriores pertenece.

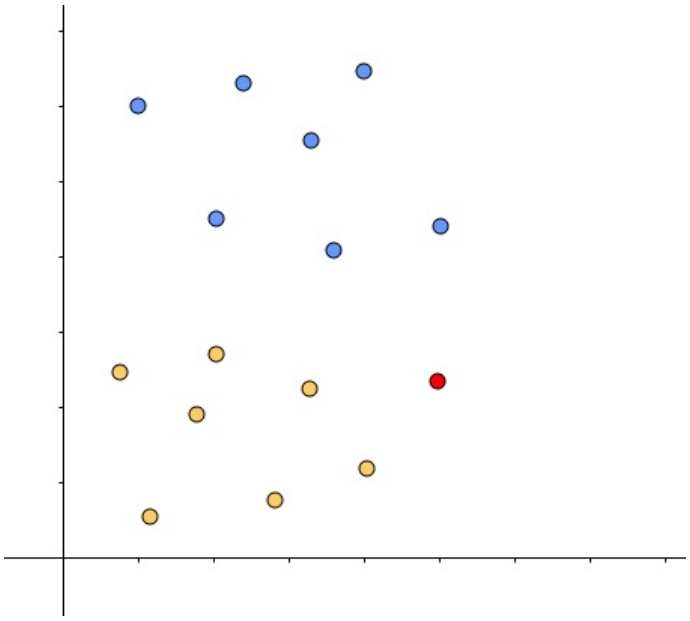


Figura 2.2: Cada uno de los colores representa una clase, mientras que el punto rojo representa un nuevo individuo.

La idea central es definir un problema de optimización basado en los vectores que conocemos la clase de pertenencia. La clasificación binaria se realiza frecuentemente utilizando la función de valores reales, $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$, de manera que a los datos de entrada $x = (x_1, ..., x_n)'$ se asigna a la clase positiva si $f(x) \geq 0$ y, en caso contrario, la clase negativa. Consideramos el caso donde $f(x)$ es una función lineal de x, de modo que se puede escribir como

$$f(x) = \langle w \cdot x \rangle + b = \sum_{i=1}^{n} w_i x_i + b,$$

donde $(w, b) \in \mathbb{R}^n \times \mathbb{R}$ son los parámetros que controlan la función. La metodología de aprendizaje implica que estos parámetros deben aprenderse de los datos, ya que dicho hiperplano tiene que separar los datos iniciales. Para construir el clasificador aplicamos la función signo al valor devuelto por f,

$$h(x) = sgn(f(x)),$$

usaremos el convenio de que sgn(0)=1.

Se utiliza la función signo como clasificador debido a que un nuevo vector x^* se clasifica en la clase 1 si $f(x^*) = \langle w \cdot x^* \rangle + b > 0$, mientras que se clasifica en la clase -1 si $f(x^*) = \langle w \cdot x^* \rangle + b < 0$. Además, la cercanía o lejanía de x^* al hiperplano puede ayudar intuitivamente a decidir cómo de buena ha sido nuestra elección de clasificación. De este modo, si $f(x^*)$ es un número lejano a 0, la elección tiene más garantía de ser acertada que si fuese cercano a 0.

Una interpretación geométrica de este tipo de hipótesis es que el espacio de entrada X se divide en dos partes, el límite de separación entre las dos clases viene dado por $f(x) = \langle w \cdot x \rangle + b = 0$ y este borde es un hiperplano afín en el caso de un separador lineal. Un hiperplano es un subespacio afín de dimensión $n - 1$ que divide el espacio en dos espacios medios que corresponden a las entradas de las dos clases distintas. El vector w define una dirección perpendicular al hiperplano, mientras que al variar el valor de b mueve el hiperplano paralelo a sí mismo. En particular, se necesita una representación que implique $n + 1$ parámetros libres, si se quiere representar todos los posibles hiperplanos en $\mathbb{R}^n$.

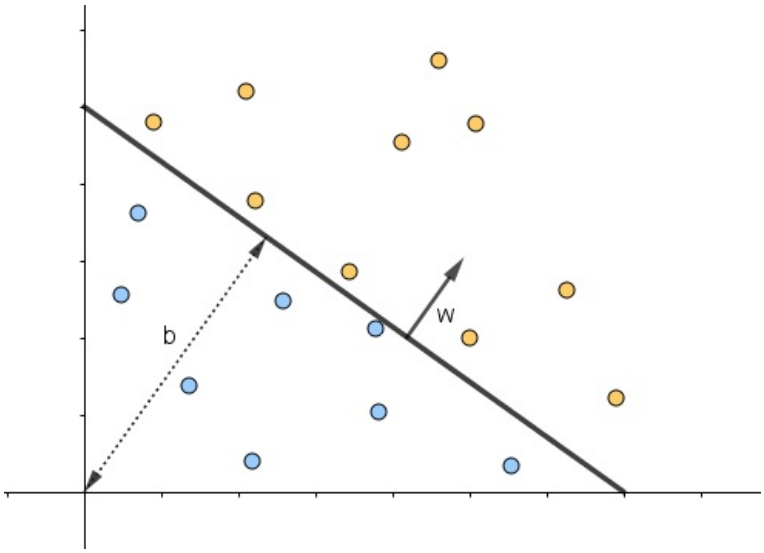


Figura 2.3: La línea representa el hiperplano de separación entre ambas clases, $f(x)$.

Nos referiremos a las cantidades w y b como el vector de ponderación y el sesgo, términos tomados de la literatura de redes neuronales. A veces $-b$ se reemplaza

por θ, una cantidad conocida como umbral. Tanto los estadísticos como los investigadores de redes neuronales han utilizado con frecuencia este tipo simple de clasificador, denominándolos respectivamente discriminadores lineales y perceptrones.

Perceptrón

El primer algoritmo iterativo para el aprendizaje de clasificaciones lineales es el procedimiento propuesto por Frank Rosenblatt en 1956 para el perceptrón. Es un procedimiento 'en línea' y basado en errores, que comienza con un vector inicial de peso w_0 (normalmente $w_0 = 0$, el vector nulo) y lo adapta cada vez que un punto de entrenamiento se clasifica erróneamente por los pesos actuales. El algoritmo actualiza directamente el vector de ponderación y el sesgo.

Algoritmo del Perceptrón

Función Perceptrón(S): un hiperplano
$w_0 \leftarrow 0$; $t \leftarrow 0$
repetir
  para i desde 1 hasta n hacer
    si $y_i\langle w_t, x_i\rangle \leq 0$ entonces
      $w_{t+1} \leftarrow w_t + y_i x_i$
      $t \leftarrow t + 1$
    fin si
  fin para
hasta que no haya errores en una iteración
retorna w

Se garantiza que este procedimiento converge siempre que exista un hiperplano que clasifique correctamente los datos de entrenamiento. En este caso, decimos que los datos son linealmente separables. Si no existe tal hiperplano, se dice que los datos no son separables.

Si ambas clases son linealmente separables, en definitiva, si es posible encontrar un hiperplano que separe los ejemplos de cada una de las clases sin errores, el problema planteado tiene múltiples soluciones ya que habrá muchos hiperplanos

capaces de separar correctamente los ejemplos.

Suponiendo el conjunto S linealmente separable, la idea central de SVM es que, además de separar las muestras de cada clase, es necesario que el hiperplano corta "justo en el medio", es decir, el hiperplano que esté más alejado de los ejemplos de ambas clases , que defina una frontera "más ancha". Esto va a ser la principal diferencia con respecto al algoritmo del perceptrón que selecciona el primer hiperplano que clasifica bien todos los ejemplos, sin tener en cuenta otros factores.

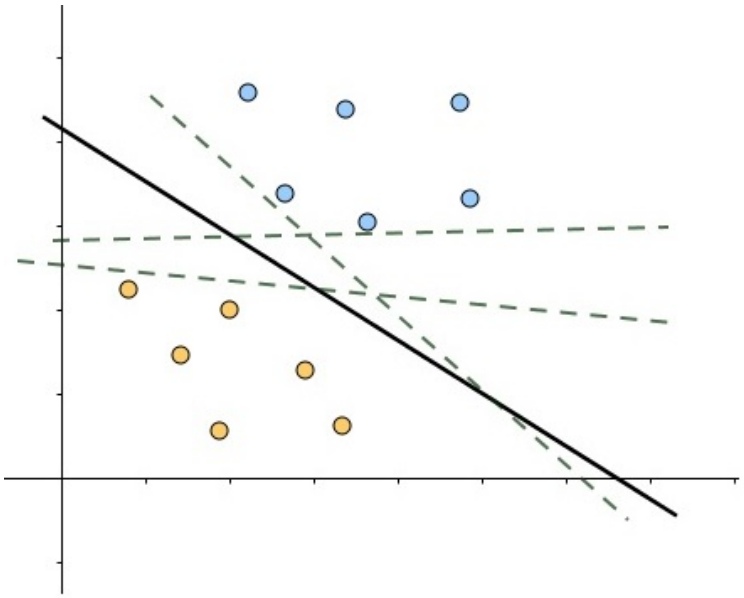


Figura 2.4: Para una misma muestra S que sea linealmente separable, existen muchos hiperplanos capaces de separa las clases. Sin embargo, hay uno que va a ser mejor, que está más distanciado de ambas clases. En este caso, el hiperplano que mejor separa las clases lo representa la línea continua negra.

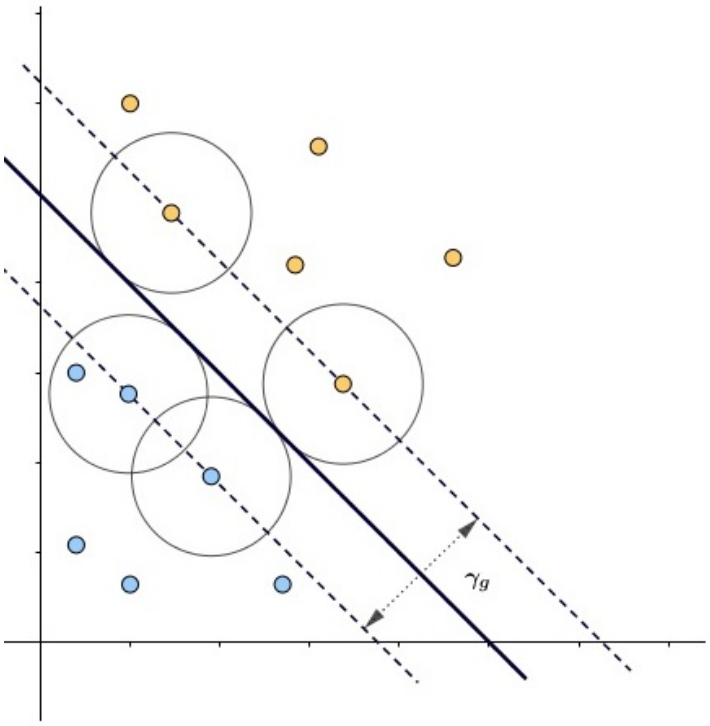


Figura 2.5: Para una misma muestra S que sea linealmente separable, existen muchos hiperplanos capaces de separa las clases. Sin embargo, hay uno que va a ser mejor, que está más distanciado de ambas clases. En este caso, el hiperplano que mejor separa las clases lo representa la línea continua negra.

Para definir formalmente esta noción, de hiperplano con la frontera más ancha, necesitamos el concepto de margen que puede entenderse de dos formas diferentes y válidas, es decir, va a haber dos tipos de márgenes relacionados entre sí:

1. **Margen funcional** Este margen hace referencia a la menor diferencia entre aplicar la función f a los ejemplos de la clase positiva y negativa,
$$\gamma_f = \min_{+}(f(x_+)) - \max_{-}(f(x_-))$$
2. **Margen geométrico** Va a ser la distancia entre los ejemplos de ambas clases. Esta distancia la podemos calcular como la suma de la distancia del hiperplano definido por w y b al ejemplo más próximo de cada clase, es decir,
$$\gamma_g = \min_{+}(d(w, b; x_+)) - \max_{-}(d(w, b; x_-))$$

Definición 5. *Definimos el margen de un ejemplo* (x_i, y_i) *con respecto a un hiperplano* (w, b) *como la cantidad*

$$\gamma_i = y_i(\langle w \cdot x_i \rangle + b),$$

donde $\gamma_i > 0$ *implica que el ejemplo* (x_i, y_i) *está clasificado correctamente.*

La distribución del margen de un hiperplano (w, b) con respecto a un conjunto de entrenamiento S es la distribución de los márgenes de los ejemplos en S. A veces nos referimos al mínimo de la distribución del margen como el margen de un hiperplano (w, b) con respecto a un conjunto de entrenamiento S. En ambas definiciones si reemplazamos el margen funcional por el margen geométrico obtenemos la cantidad equivalente para la función lineal normalizada $\left(\frac{1}{\|w\|}w, \frac{1}{\|w\|}b\right)$, que mide las distancias euclidianas de los puntos del límite de decisión en el espacio de entrada. Finalmente, el margen de un conjunto de entrenamiento S es el margen geométrico máximo sobre todos los hiperplanos. Un hiperplano que se da cuenta de este máximo se conoce como hiperplano de margen máximo. El tamaño de su margen será positivo para un conjunto de entrenamiento linealmente separable.

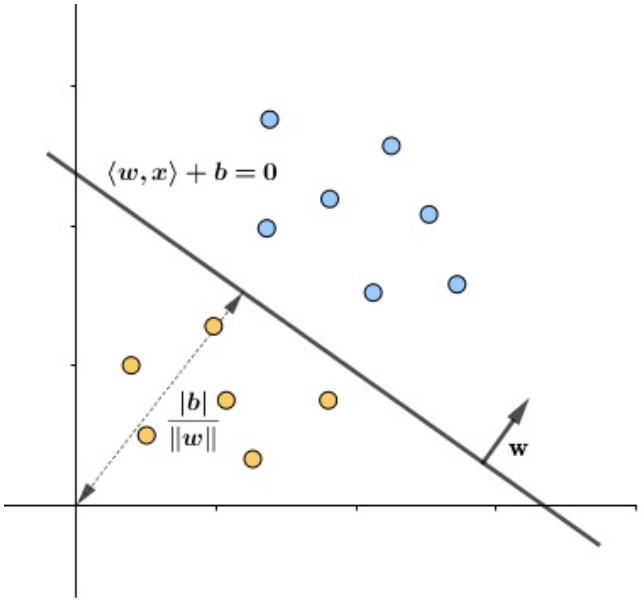


Figura 2.6: Hiperplano que es perpendicular a w y cuya distancia al origen depende de b y de $\|w\|$

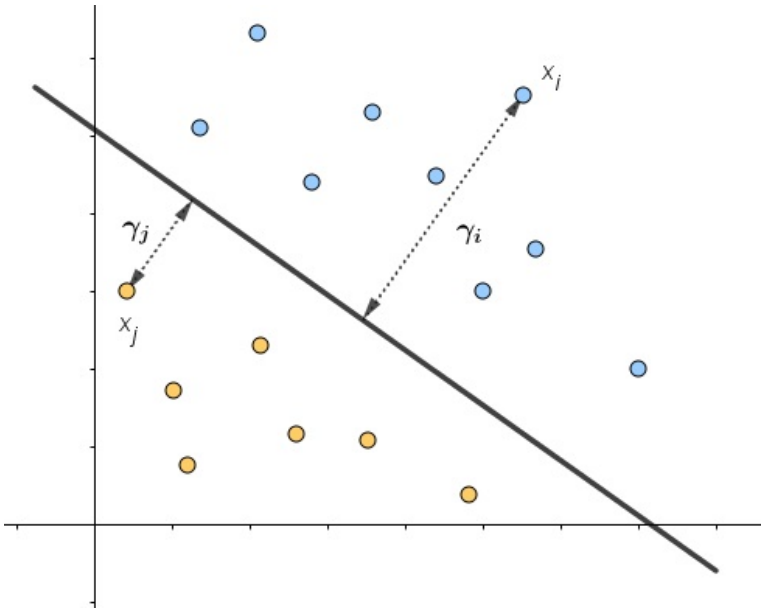


Figura 2.7: Margen geométrico en dos puntos con respecto a un hiperplano en dos dimensiones.

Teorema 1. *Sea S un conjunto de entrenamiento no trivial y sea*

$$R = \max_{1 \leq i \leq l} \|x_i\|.$$

Supongamos que existe un vector w_{opt} tal que $\|w_{opt}\| = 1$ y

$$y_i(\langle w_{opt} \cdot x_i \rangle + b_{opt}) \geq \gamma$$

para $1 \leq i \leq l$. Entonces, el número de errores cometidos por el algoritmo del perceptrón en línea en el conjunto S es como máximo

$$\left(\frac{2R}{\gamma}\right)^2$$

∗ **Demostración**:
Para el análisis, se aumentan los vectores de entrada mediante una coordenada adicional con valor R. Se va a denotar el nuevo vector por $\hat{x_i} = (x_i', R)'$, donde se denota x' como la traspuesta de x. De manera similar, se agrega una coordenada adicional al vector de ponderación w incorporando el sesgo b, para formar el vector de peso aumentado $\hat{w} = (w', \frac{b}{R})'$. El algoritmo comienza con un vector de peso

aumentado $\hat{w_0} = 0$ que se va actualizando en cada error. Sea $\hat{w}_{t-1}$ el vector de peso aumentado antes del t-ésimo error. La actualización t-ésima se realiza cuando

$$y_i\langle\hat{w}_{t-1}\cdot\hat{x_i}\rangle = y_i(\langle w_{t-1}\cdot x_i\rangle + b_{t-1}) \leq 0$$

donde $(x_i, y_i) \in S$ es el punto mal clasificado por $\hat{w_{t-1}} = (w'_{t-1}, \frac{b_{t-1}}{R})'$. La actualización es la siguiente:

$$\hat{w_t} = (w'_t, \frac{b_t}{R})' = (w'_{t-1}, \frac{b_{t-1}}{R})' + \eta y_i(x'_i, R)' = \hat{w}_{t-1} + \eta y_i\hat{x_i},$$

donde se ha usado el hecho de que

$$\frac{b_t}{R} = \frac{b_{t-1}}{R} + \eta y_i R$$

ya que $b_t = b_{t-1} + \eta y_i R^2$.

La derivación

$$\langle\hat{w_t}\cdot\hat{w}_{opt}\rangle = \langle\hat{w}_{t-1}\cdot\hat{w}_{opt}\rangle + \eta y_i\langle\hat{x_i}\cdot\hat{w}_{opt}\rangle \geq \langle\hat{w}_{t-1}\cdot\hat{w}_{opt}\rangle + \eta\gamma$$

implica (por inducción) que

$$\langle\hat{w_t}\cdot\hat{w}_{opt}\rangle \geq t\eta\gamma.$$

De forma similar, se tiene

$$\begin{aligned}\|\hat{w_t}\|^2 &= \|\hat{w}_{t-1}\|^2 + 2\eta y_i\langle\hat{w}_{t-1}\cdot\hat{x_i}\rangle + \eta^2\|\hat{x_i}\|^2 \\ &\leq \|\hat{w}_{t-1}\|^2 + \eta^2\|\hat{x_i}\|^2 \\ &\leq \|\hat{w}_{t-1}\|^2 + \eta^2(\|x_i\|^2 + R^2) \\ &\leq \|\hat{w}_{t-1}\|^2 + 2\eta^2R^2,\end{aligned}$$

lo que implica que

$$\|\hat{w_t}\|^2 \leq 2t\eta^2R^2.$$

Las dos desigualdades combinadas dan la relación

$$\|\hat{w}_{opt}\|\sqrt{2t}\eta R \geq \|\hat{w}_{opt}\|\|\hat{w_t}\| \geq \langle\hat{w_t}, \hat{w}_{opt}\rangle \geq t\eta\gamma,$$

que juntos implican el límite

$$t \leq 2\left(\frac{R}{\gamma}\right)^2 \|\hat{w}_{opt}\|^2 \leq \left(\frac{2R}{\gamma}\right)^2,$$

ya que $b_{opt} \leq R$ para una separación no trivial de los datos y entonces,

$$\|\hat{w}_{opt}\|^2 \leq \|w_{opt}\|^2 + 1 = 2.$$

□

El teorema demuestra que el algoritmo converge en un número finito de iteraciones siempre que su margen sea positivo. Simplemente iterando varias veces en el mismo conjunto S, después de una serie de errores limitados por $\left(\frac{2R}{\gamma}\right)^2$ el algoritmo perceptron encontrará un hiperplano de separación y, por lo tanto, se detendrá, siempre que exista uno.

En los casos donde los datos no son separables linealmente, el algoritmo no convergerá. Sin embargo, existe un teorema similar al de Novikoff, que limita el número de errores cometidos durante una iteración. Utiliza una medida diferente de la distribución del margen utilizando los márgenes obtenidos por los puntos de entrenamiento distintos de los más cercanos al hiperplano.

Definición 6. *Fijado un valor $\gamma > 0$, podemos definir la variable de holgura de margen de un ejemplo (x_i, y_i) con respecto al hiperplano (w, b) y el margen objetivo γ como*

$$\xi((x_i, y_i), (w, b), \gamma) = \xi_i = \text{máx}(0, \gamma - y_i(\langle w \cdot x_i \rangle + b)).$$

Informalmente, esta cantidad mide cuánto falla un punto que tiene margen γ desde el hiperplano. Si $\xi_i > \gamma$, entonces x_i está mal clasificado por (w, b). La norma $\|\xi\|_2$ mide la cantidad por el que el conjunto de entrenamiento no obtiene el margen γ y tiene en cuenta las clasificaciones erróneas de los datos de entrenamiento.

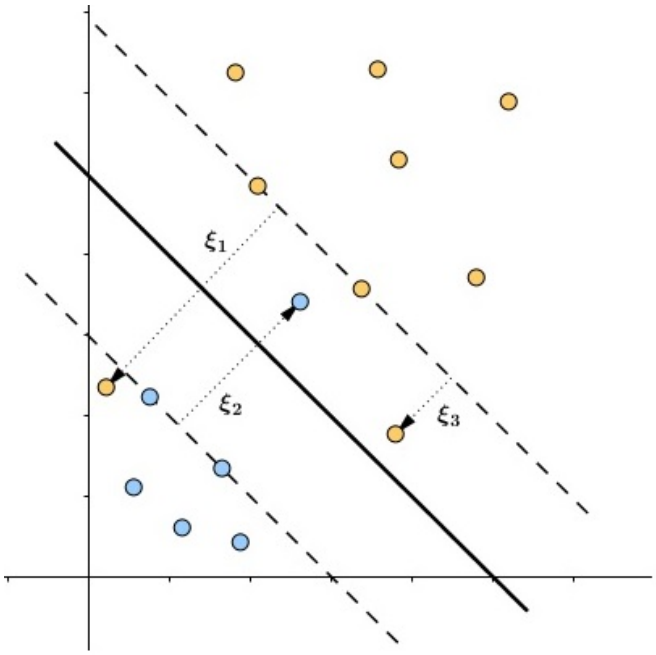


Figura 2.8: Muestra el tamaño de las variables de holgura para los puntos mal clasificados para un hiperplano con norma de unidad. Todos los demás puntos de la figura tienen su variable de holgura cero, ya que tienen un margen (positivo) mayor que γ

Teorema 2. *Sea S un conjunto de entrenamiento no trivial sin ejemplos duplicados, con $\|x_i\| \leq R$. Sea (w, b) cualquier hiperplano con $\|w\| = 1$, con $\gamma > 0$ y definimos*

$$D = \sqrt{\sum_{i=1}^{l} \xi_i^2} = \sqrt{\sum_{i=1}^{l} \xi((x_i, y_i), (w, b), \gamma)^2}.$$

Entonces, el número de errores en la primera ejecución del bucle for del algoritmo perceptrón en el conjunto S está limitado por

$$\left(\frac{2(R+D)}{\gamma}\right)^2.$$

∗ **Demostración**:
La prueba define un espacio de entrada ampliado parametrizado por Δ en el que hay un hiperplano con margen γ que tiene la misma funcionalidad que $(w', b)'$ en datos no vistos. Luego, se puede aplicar el Teorema 1 en el espacio extendido. Finalmente, optimizando la elección de Δ dará el resultado. El espacio de entrada extendido tiene una coordenada adicional para cada ejemplo de entrenamiento. Las nuevas entradas por ejemplo x_i son todas cero excepto para el valor *Delta* en la i-ésima coordenada adicional. Se denota $\tilde{x}_i$ al vector extendido y $\tilde{S}$ el conjunto de entrenamiento correspondiente. Ahora, se extiende w con el valor $\frac{y_i \xi_i}{\Delta}$ en la i-ésima entrada adicional para dar el vector $\tilde{w}$. Observar que

$$y_i(\langle \tilde{w} \cdot \tilde{x}_i \rangle + b) = y_i(\langle w \cdot x_i \rangle + b) + \xi_i \geq \gamma,$$

mostrando que $(\tilde{w}', b)'$ tiene un margen γ en $\tilde{S}$. Notar, sin embargo, que

$$\|\tilde{w}\| = \sqrt{1 + \frac{D^2}{\Delta^2}},$$

de modo que el margen geométrico $\tilde{\gamma}$ se reduce por este factor. Como los ejemplos de entrenamiento extendido tienen entradas distintas de cero en diferentes coordenadas, ejecutar el algoritmo perceptron para el primer bucle forzado en $\tilde{S}$ tiene el mismo efecto que ejecutarlo en S y se puede enlazar el número de errores por el Teorema 1 por

$$\left(\frac{2\tilde{R}}{\tilde{\gamma}}\right)^2 = \frac{4(R^2 + \Delta^2)(1 + \frac{D^2}{\Delta^2})}{\gamma^2}.$$

El límite es optimizado eligiendo $\Delta = \sqrt{RD}$ dando el resultado requerido.

□

Es importante tener en cuenta que el algoritmo de perceptrón funciona al agregar ejemplos de entrenamiento positivo mal clasificados o al restar los negativos de clase errónea a un vector de peso arbitrario inicial. Sin pérdida de generalidad, hemos supuesto que el vector de peso inicial es el vector cero, por lo que la hipótesis final será una combinación lineal de los puntos de entrenamiento,

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i,$$

donde, dado que el signo del coeficiente x_i viene dado por la clasificación y_i, los α_i son valores positivos proporcionales al número de veces que la clasificación errónea de x_i ha ocasionado que se actualice el peso. Aquellos ejemplos (puntos) más difíciles de clasificar tendrán valores más altos de α_i, mientras que los ejemplos bien clasificados valdrá 0 en todas las iteraciones (esta cantidad a veces se conoce como la fuerza de inserción del patrón x_i). Una vez que se ha fijado un conjunto de entrenamiento S, se puede pensar en el vector α como una representación alternativa de las hipótesis en coordenadas duales. Esta expansión, sin embargo, no es única ya que diferentes α pueden corresponder a la misma hipótesis w. Intuitivamente, uno también puede considerar α_i como una indicación de la dificultad de clasificar el ejemplo x_i. En el caso de los datos no separables, los coeficientes de los puntos mal clasificados crecen indefinidamente. La función de decisión se puede reescribir en coordenadas duales de la siguiente manera:

$$\begin{aligned} h(x) &= sgn(\langle w, x\rangle + b) \\ &= sgn\left(\left\langle \sum_{i=1}^{l} \alpha_i y_i x_i \cdot x \right\rangle + b\right) \\ &= sgn\left(\sum_{i=1}^{l} \alpha_i y_i \langle x_i \cdot x\rangle + b\right), \end{aligned}$$

y el algoritmo del perceptrón también se puede expresar completamente en esta forma dual, lo que aporta buenas propiedades ya que los puntos difíciles de aprender se pueden utilizar para clasificar los datos de acuerdo con su contenido de información.

El problema de aprender un hiperplano que separa dos conjuntos de puntos (separables) es un problema planteado, en el sentido de que, en general, existen varias soluciones diferentes. El peligro de los problemas mal planteados es que no todas las soluciones pueden ser igualmente útiles. De esta forma, podemos elegir no aprender cualquier regla que separe correctamente las dos clases, sino la que está a una distancia máxima de los datos. Este es el hiperplano que realiza el margen máximo.

Para tratar de maximizar cualquiera de estos márgenes hay que maximizar uno de ellos y mantener el otro fijo. Lo más habitual es maximizar el margen geométrico manteniendo fijo el funcional. De modo que consideramos sólo los hiperplanos

canónicos, donde os parámetros w y b están restringidos por las siguientes condiciones que sirven para fijar el margen funcional:

$$\begin{aligned} &\min_{+}(\langle w, x_+\rangle + b = +1,) \\ &\max_{-}(\langle w, x_-\rangle + b = -1.) \end{aligned}$$

Con ello, podemos calcular el margen geométrico γ_g para un hiperplano canónico cualquiera:

$$\begin{aligned} \gamma_g &= \min_{+}(d(w, b; x_+)) - \min_{-}(d(w, b; x_-)) \\ &= \min_{+}\left(\frac{|\langle w, x_+\rangle + b|}{\|w\|}\right) - \min_{-}\left(\frac{|\langle w, x_-\rangle + b|}{\|w\|}\right) \\ &= \frac{|+1|}{\|w\|} + \frac{|-1|}{\|w\|} \\ &= \frac{2}{\|w\|}. \end{aligned}$$

En definitiva, para maximizar el margen geométrico debemos minimizar la norma de w, luego tenemos un problema de optimización de la siguiente forma:

$$\begin{aligned} \min \quad & \|w\| \\ \text{s.a.} \quad & (\langle w, x_i\rangle + b) \geq +1, \quad \forall x_i \in +1, \\ & (\langle w, x_i\rangle + b) \leq -1, \quad \forall x_i \in -1. \end{aligned} \tag{2.1}$$

cuya solución será el hiperplano de margen geométrico máximo que clasifica correctamente todos los ejemplos[1].

Para simplificar y facilitar la resolución del problema anterior sustituiremos la función objetivo ($\|w\|$) por su equivalente ($\langle w, w\rangle$) y se incluirá a constante multipicativa $\left(\frac{1}{2}\right)$. Además, unificaremos las expresiones de las restricciones y obtendremos una formulación equivalente al anterior problema de mínimo que es el siguiente:

$$\begin{aligned} \min \quad & \frac{1}{2}\langle w, w\rangle \\ \text{s.a.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1, \quad i = 1, ..., n. \end{aligned} \tag{2.2}$$

[1]Para resolver dicho problema se van a aplicar métodos conocidos de optimización de funciones que se desarrollan más adelante en la sección 2.3.

La función langrangiana asociada a dicho problema es:

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

siendo los $\alpha_i \geq 0$ las variables duales que se denominan multiplicadores de Lagrange. El signo negativo del sumatorio viene dado al invertir las restricciones de $\leq$ a $\geq$. El problema dual se determina diferenciando la función lagrangiana respecto a las variables primales (w y b),

$$\frac{\partial L(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0,$$
$$\frac{\partial L(w, b, \alpha)}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0.$$

A partir de las ecuaciones anteriores obtenemos la relación entre variables primales y duales, que viene dada por,

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i.$$

Mediante esta relación podemos pasar al problema dual sustituyéndola en la función lagrangiana,

$$\begin{aligned} L(w, b, \alpha) &= \frac{1}{2} \langle w, w \rangle - \sum_{i=1}^{n} \alpha_i [y_i(\langle w, x_i \rangle + b) - 1] \\ &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i x_j \rangle - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i x_j \rangle \\ &- \sum_{i=1}^{n} \alpha_i y_i b + \sum_{i=1}^{n} \alpha_i \\ &= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i x_j \rangle + \sum_{i=1}^{n} \alpha_i \end{aligned}$$

El problema dual equivalente es el siguiente:

$$\begin{aligned} \text{máx} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_{i=1}^{n}\alpha_i \\ \text{s.a.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \\ & \alpha_i \geq 0 \quad i = 1, ..., n. \end{aligned} \tag{2.3}$$

Los productos escalares que aparecen entre los ejemplos de entrenamientos se pueden sustituir por la aplicación de una función kernel. Además, las características de las funciones kernel nos garantizan que el problema dual siga teniendo solución (debe ser simétrica y definida positiva). El valor de la variables b, que no aparece en el problema dual, se calculará una vez obtenido el valor óptimo de w, que lo denotaremos por w^*. El valor óptimo de b (b^*) va a venir dado por las restricciones obtenidas al fijar el margen funcional, es decir,

$$b^* = -\ \frac{\text{máx}_-(\langle w^*, x_-\rangle) + \text{mín}_+(\langle w^*, x_+\rangle)}{2} \tag{2.4}$$

A partir de los datos obtenidos del problema dual de w^*,b^* podemos construir el clasificador que buscamos utilizando el signo al aplicar la función f a cualquier x:

$$h(x) = sgn(f(x)) = sgn\left(\sum_{i=1}^{n}\alpha_i^* y_i \langle x_i, x\rangle + b^*\right). \tag{2.5}$$

La solución, en particular, va a ser una combinación lineal de aquellos datos de entrenamiento que tengan un multiplicador de Lagrange distinto de 0. Dichos ejemplos los denominaremos *vectores soporte*, que en el caso separable van a ser los que estén justo en el margen de cada clase de forma que el resto (es decir, los ejemplos que tengan multiplicador cero) podríamos eliminarlos antes del aprendizaje y se generaría el mismo hiperplano separador.

2.1.2. Margen blando

El problema que se ha estudiado hasta el momento, en el que los ejemplos son linealmente separables, tiene un escaso interés desde el punto de vista práctico ya que es muy difícil encontrar aplicaciones reales con este tipo de ejemplos. La

única posibilidad es generar, a partir de ciertos kernel, espacios de características de muy alta dimensión donde los datos fueran más fácilmente separables, aunque en estos casos es muy posible que nos sobreajustemos a los datos de entrenamiento al emplear un espacio de hipótesis excesivamente rico. En definitiva, debemos generalizar el problema de forma que determinados ejemplos puedan estar mal clasificados por el hiperplano elegido (el problema de encontrar el hiperplano que clasifique de manera correcta el mayor número de ejemplos cuando el conjunto de entrenamiento no es linealmente separable).

Por otro lado, permitir que algunos ejemplos estén mal clasificados se traduce en que las restricciones de dichos ejemplos puedan ser violadas, es decir que no se verifiquen siempre. Para ello, en cada restricción original se introduce una variable de holgura, que denotaremos por ξ_i y nos va a indicar el número de veces que se viola la condición de clasificar bien dicho ejemplo. De está forma, las restricciones originales las vamos a sustituir por las siguientes:

$$y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \quad i = 1, ..., n.$$

Así, ξ_i valdrá cero cuando el ejemplo esté en la zona definida por su margen, y tomará un valor positivo (mayor que cero) en caso contrario. Luego, dicha cantidad refleja la diferencia funcional entre el valor que toma el ejemplo y el que debería tomar para estar en la región de su clase determinada por el margen. Es decir, el ejemplo debe estar bien clasificado y más allá del margen. Para la función objetivo, además de maximizar el margen debemos añadirle los errores que está cometiendo el hiperplano y nos indique el coste de la solución. En definitiva, el problema de optimización para el caso general no separable linealmente viene dado por:

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n}\xi_i, \\ \text{s.a.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \quad i = 1, ..., n, \\ & \xi_i \geq 0 \quad i = 1, ..., n. \end{aligned} \tag{2.6}$$

La constante C permite regular el grado de sobreajuste que permitimos a la hipótesis generada (es decir, en qué grado influye dicho término en relación con la minimización de la norma). Va a ser uno de los parámetros claves al aplicar una SVM. La función lagrangiana asociada va a tener dos grupos de multiplicadores

de Lagrange, uno para cada conjunto de restricciones:

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

Para obtener las relaciones entre las variables primales y duales, como en el caso linealmente separable, calculamos las derivadas parciales respecto a las variables del problema primal que en este caso son w,b y ξ e igualamos a cero:

$$\begin{aligned}
\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial w} &= w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0,\\
\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial b} &= \sum_{i=1}^{n}\alpha_i y_i = 0,\\
\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0.
\end{aligned}$$

De manera que las relaciones definitivas entre ambos grupos de variables vienen dadas por las siguientes restricciones:

$$w = \sum_{i=1}^{n}\alpha_i y_i x_i,$$

$$C = \alpha_i + \beta_i.$$

Introduciendo estas relaciones de nuevo obtenemos el problema dual definitivo que queda,

$$\begin{aligned}
\text{máx} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i,x_j\rangle + \sum_{i=1}^{n}\alpha_i\\
\text{s.a.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0,\\
& 0 \le \alpha_i \le C \quad i = 1,...,n.
\end{aligned} \tag{2.7}$$

Lo que se asemeja mucho al caso separable, con la diferencia de que los valores de los multiplicadores de Lagrange asociado a las restricciones relativas a la clasificación de cada ejemplo, α_i, están acotados superiormente con la constante C.

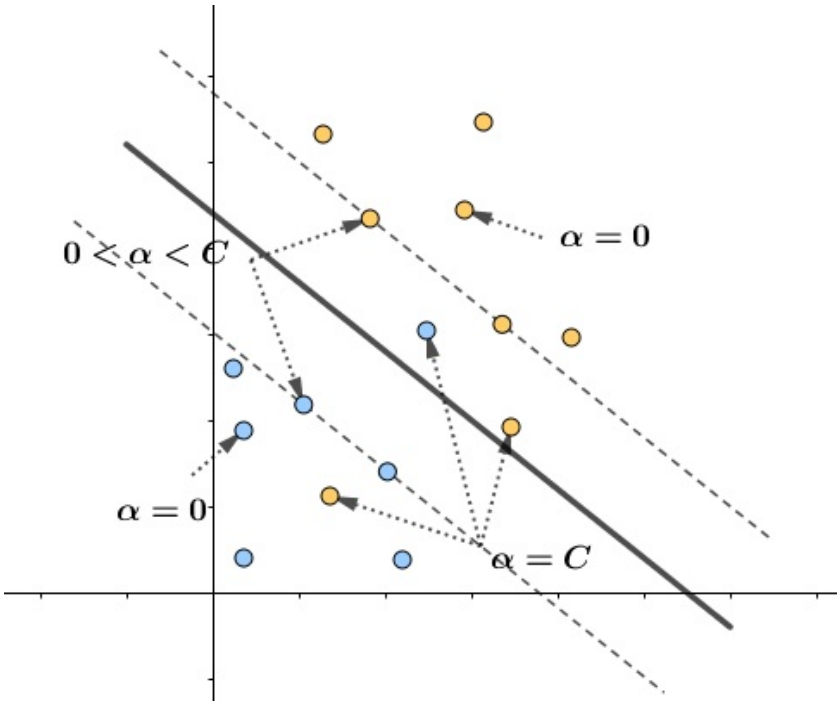


Figura 2.9: Valores de los multiplicadores de Lagrange. El valor α de cada ejemplo depende de la posición en la que se encuentre respecto del hiperplano y la frontera de margen de cada clase.

Para un problema de clasificación de clase múltiple, el dominio de salida es $Y = \{1, 2, .., m\}$. La generalización de las máquinas de aprendizaje lineal para el caso de m-clases es sencilla: a cada una de las m-clases se le asocia un vector de ponderación y un sesgo, (w_i, b_i), con $i \in 1, .., m$ y la función de decisión está dada por

$$c(x) = \arg \max_{1 \leq i \leq m} (\langle w_i \cdot x \rangle + b_i).$$

Geométricamente, esto equivale a asociar un hiperplano a cada clase, y a asignar un nuevo punto x a la clase cuyo hiperplano está más alejado de él. El espacio de entrada se divide en m regiones conectadas y convexas. Existen algoritmos para aprender los m hiperplanos simultáneamente a partir de datos, y son extensiones de los procedimientos básicos descritos anteriormente.

2.2. Métodos Kernel

Los métodos kernel es una familia de algoritmos cuyo elemento elemento común y pieza fundamental en todos es la función kernel, su utilidad en el análisis de datos reside en la representación de la información. Este tipo de métodos presentan la ventaja de que son aplicables a cualquier tipo de datos, además se pueden aplicar algoritmos lineales y obtener con ellos soluciones no lineales. Las representaciones Kernel ofrecen una solución alternativa al proyectar los datos en un espacio de características de alta dimensión para aumentar la potencia de cálculo de las máquinas de aprendizaje lineal.

La ventaja de utilizar las máquinas en la representación dual deriva del hecho de que, en esta representación, el número de parámetros ajustables no depende del número de atributos que se utilizan. Al reemplazar el producto interno con una función de kernel elegida apropiadamente, se puede realizar implícitamente una función no lineal a un espacio de características de alta dimensión sin aumentar el número de parámetros ajustables, siempre que el núcleo calcule el producto interno de los vectores de características correspondientes a las dos entradas. Por tanto, el problema consiste en elegir un kernel adecuado para la SVM. El principal interés de los kernel, en el contexto de SVM, es que lo que se ha visto en el caso de separación lineal también se aplica fácilmente a la separación no lineal mediante su uso.

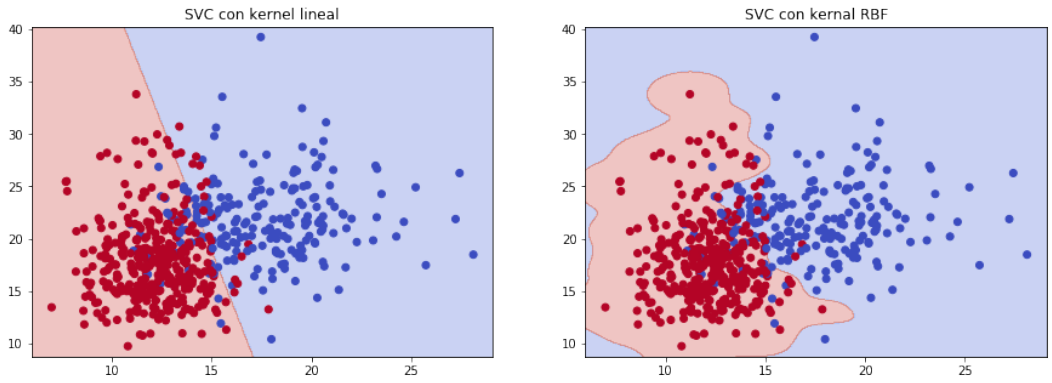


Figura 2.10: Para la nube de puntos representada el kernel RBF va a ser más adecuado que el lineal para SVM.

Dado un conjunto de objetos $S = \{x_1, ..., x_l\}$, cada $x_i \in X$ representa un objeto diferente que puede ser una proteína, imagen, cliente, etc. Una manera de representar los datos consiste en definir para cada objeto $x_i \in X$ una representación $\phi(x_i) \in F$ que puede consistir en un vector de números reales, una secuencia de longitud la dimensión del espacio de 0's y 1's, una cadena de caracteres de tamaño fijo, etc; de forma que se podría representar S como el conjunto de representaciones de cada objeto, $\phi(S) = \{\phi(x_1), ..., \phi(x_l)\}$.

Con frecuencia se busca identificar el conjunto más pequeño de características que que contenga la información esencial de los atributos originales, lo que se conoce como reducción de dimensionalidad,

$$\mathbf{x} = (x_1, .., x_l) \mapsto \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), .., \phi_d(\mathbf{x})), con \;\; d < l,$$

lo que puede ser beneficioso, ya que tanto el rendimiento computacional como el de generalización puede degradarse a medida que aumenta el número de características, cuanto mayor sea dicho conjunto, más probable es que la función que se va a aprender se pueda representar utilizando una máquina de aprendizaje estandarizada y, además, puede crear problema de sobreajuste.

Para aprender relaciones no lineales con una máquina lineal necesitamos seleccionar un conjunto de características no lineales y reescribir los datos en la nueva representación. Esto es equivalente a aplicar una función no lineal fija de los datos a un espacio de características, en el que se puede usar la máquina lineal. Entonces, el conjunto de hipótesis que consideramos serán funciones del tipo

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

donde $\phi : X \to F$ es una función no lineal del espacio de entrada a un espacio de característica. Esto significa que se construyen máquinas no lineales en dos pasos: primero, una asignación fija no lineal que transforma los datos en un espacio de características F y, posteriormente, se usa una máquina lineal para clasificarlos en el espacio de características.

Las máquinas de aprendizaje lineal se pueden expresar en una representación dual. Una consecuencia importante de la representación dual es que la dimensión del espacio de características no tiene por qué afectar en el cálculo. Con dicha representación, la hipótesis se puede expresar como una combinación lineal de los puntos

de entrenamiento, de modo que la regla de decisión se puede evaluar utilizando solo productos internos entre el punto de prueba y los puntos de entrenamiento, es decir:

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

De manera que si tenemos una forma de calcular el producto interno $\langle \phi(x_i) \cdot \phi(x) \rangle$ en el espacio de características directamente en función de los puntos de entrada originales, es posible fusionar los dos pasos necesarios para construir una máquina de aprendizaje no lineal. A este método de cálculo directo lo denominaremos función de kernel. Los métodos kernel son algoritmos que procesan la información representada mediante el uso de funciones kernel, es decir, reciben la información a procesar en forma de matrices calculadas mediante una función kernel.

La 'kernelización' de algoritmos consiste en reemplazar el producto escalar de la formulación original por la evaluación de una función kernel. Con este truco, se puede extender la aplicación de algoritmos clásicos en espacios vectoriales dotados de un producto escalar a cualquier otro tipo de datos siempre que se pueda definir un kernel. Esto lo podemos aplicar a muchos campos de aplicaciones a día de hoy como en el ámbito de la biología [45].

Como hemos citado anteriormente, el perceptrón [46] forma parte de la familia de los clasificadores lineales, algoritmos que producen una función lineal que se pueden utilizar para diferenciar entre ejemplos de dos clases. Este tipo de algoritmos se adaptan a muchos problemas reales como decidir si un paciente tiene o no una enfermedad, si se le concede un préstamo bancario a un cliente, etc. La idea es que la función no lineal que representa la solución que buscamos en el espacio de entrada es una función lineal en algún espacio de características.

Los métodos kernel representan la información mediante una función k, $k : X \times X \to \mathbb{R}$, que calcula la similitud de cada par de objetos del conjunto de entrada X. De esta forma, la representación del conjunto S viene dado por la matriz K de números reales con dimensión $l \times l$, donde cada elemento de dicha matriz es el resultado de aplicar la función k a cada par de objetos del espacio de entrada, es decir, $K_{ij} = k(x_i, x_j)$. Esta matriz se conoce como la matriz kernel, y en este contexto, utilizaremos el símbolo K para denotarla.

Una de las ventajas que presentan este tipo de representación es su modularidad, es decir, que son capaces de diseñar algoritmos genéricos que manipulan la información de matrices de números reales sin la necesidad de conocer la función k que se utilice ni el tipo de objetos del dominio. Además, permiten que algoritmos de análisis relativamente sencillos encuentren relaciones complejas entre objetos.

Definición 7. *Una función kernel es una función* $k : X \times X \to \mathbb{R}$*, que asigna a cada par de objetos del espacio de entrada, X, un valor real correspondiente al producto escalar de las imágenes de dichos objetos en un espacio F, que denominaremos espacio de características, es decir,*

$$k(x, z) = \langle \phi(x), \phi(z) \rangle ,$$

donde $\phi : X \to F$*.*

Definición 8. *Una función* $k : X \times X \to \mathbb{R}$ *es simétrica si para cada par de objetos* $x, z \in X$ *se tiene que*

$$k(x, z) = k(z, x).$$

Definición 9. *Una función* $k : X \times X \to \mathbb{R}$ *es semi-definida positiva si*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

para cualquier conjunto de n objetos $x_1, ..., x_n$ *de X y cualquier conjunto de valores reales* $c_1, ..., c_n$ *con* $n > 0$

La clave de este enfoque es encontrar una función kernel que pueda evaluarse de manera eficiente. Una vez que tengamos tal función, la regla de decisión puede ser evaluada por, como máximo, l evaluaciones del kernel:

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i k(x_i, x) + b.$$

Un aspecto importante del uso de kernel es que no necesitamos conocer la función subyacente para poder aprender en el espacio de características. Además, un kernel generaliza el producto interno estándar en el espacio de entrada, ya que dicho producto interno proporciona un ejemplo de kernel haciendo que la característica trace la identidad

$$k(x, z) = \langle x \cdot z \rangle .$$

También podemos tomar la función de características mediante una transformación lineal fija $x \mapsto Ax$, para alguna matriz A. En este caso, la función kernel está dada por

$$k(x, z) = \langle Ax \cdot Az \rangle = x'A'Az = x'Bx,$$

donde, $B = A'A$ que es una matriz cuadrada, simétrica, semidefinida positiva.

Vamos a ver ahora algunos ejemplos de las funciones kernel más relevantes, cómo se contruyen y condiciones que tienen que cumplir.

2.2.1. Ejemplos de funciones Kernel

Las funciones kernel más habituales aplicables cuando $X \subseteq \mathbb{R}^n$ son:

(a) *Kernel lineal*:

$$k(x, z) = \langle x \cdot z \rangle = \sum_{i=1}^{l} x_i z_i$$

(b) *Kernel polinómico*:

Una función kernel polinómica de grado d se expresa como

$$k(x, z) = (\langle x \cdot z \rangle)^d$$

o de la siguiente forma,

$$k(x, z) = (\langle x \cdot z \rangle + c)^d$$

Corresponde a una proyección de $\phi(x)$ en un espacio de características donde cada componente $\phi_i(x)$ es un producto de componentes de x con grado menor que d, es decir, un monomio. El separador calculado a partir de este kernel es un polinomio de grado d, cuyos términos son las componentes de x. En definitiva, la función ϕ asociada a dicho kernel lleva a cada vector de entrada en un vector con todos los posibles monomios de grado d, para el primer caso, o a vectores compuestos por todos los monomios de grado menor o igual a d, en el segundo.

(c) *Kernel gaussiano*:

Se trata de una función de base radial gaussiana

$$k(x, z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$$

Corresponde a una proyección en un espacio de dimensiones finitas. El kernel gaussiano tiene la peculiaridad de que $k(x, x) = e^0 = 1 \ \forall x \in X$, luego $\|\phi(x)\| = 1$. Además, $k(x, z) > 0$ para todo $x, z \in X$ esto nos lleva a que el ángulo entre cualquier par de imágenes es menor que $\frac{\pi}{2}$, de forma que las imágenes de los vectores del espacio de entrada caen dentro de una región restringida del espacio de características.

(d) *Combinación de funciones kernel*:

Sean:

- k_1, k_2, k_3 tres funciones kernel.
- f una función de valores reales.
- ϕ una función que proyecta los vectores en otro espacio vectorial.
- B una matriz semidefinida positiva.
- p un polinomio con coeficientes positivos.
- α un número positivo

entonces, las siguientes funciones k también son funciones kernel:

1. $k(x,z) = k_1(x,z) + k_2(x,z)$
2. $k(x,z) = \alpha k_1(x,z)$
3. $k(x,z) = k_1(x,z)k_2(x,z)$
4. $k(x,z) = f(x)f(z)$
5. $k(x,z) = k_3(\phi(x),\phi(z))$
6. $k(x,z) = x^T Bz$
7. $k(x,z) = p(k_1(x,z))$
8. $k(x,z) = \exp(k_1(x,z))$

Vamos a determinar las propiedades de una función $k(x,z)$ que son necesarias para garantizar que se trate de un espacio de características del kernel. Para ello, la función debe ser simétrica y satisfacer las desigualdades que se derivan de la desigualdad de Cauchy-Schwarz,

$$\begin{aligned} k(x,z)^2 &= \langle \phi(x) \cdot \phi(z) \rangle^2 \leq \|\phi(x)\|^2 \|\phi(z)\|^2 \\ &= \langle \phi(x) \cdot \phi(x) \rangle \langle \phi(z) \cdot \phi(z) \rangle = k(x,x)k(z,z). \end{aligned}$$

Aunque estas condiciones no son suficientes para caracterizar la existencia de un espacio de características. A continuación vamos a ver el teorema de Mercer, que proporciona una caracterización de cuando una función $k(x,z)$ es un kernel.

Proposición 1. *Sea $X = \{x_1, ..., x_n\}$ un espacio de entrada finito con $k(x,z)$ una función simétrica en X. Entonces, $k(x,z)$ es una función kernel si y solo si la matriz*

$$K = (K_{ij})_{i,j=1}^n = (k(x_i, x_j))_{i,j=1}^n,$$

es semidefinida positiva (es decir, tiene autovalores no negativos).

Una generalización de un producto interno en un espacio de Hilbert al introducir una ponderación λ_i para cada dimensión,

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

de modo que el vector característica se convierte en

$$\phi(x) = (\phi_1(x), \phi_2(x), ..., \phi_n(x), ...)$$

De forma que, el teorema de Mercer da las condiciones necesarias y suficientes para que una función simétrica continua $k(x, z)$ admita tal representación

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

con λ_i no negativo, que es equivalente a $k(x, z)$ siendo un producto interno en el espacio de característica $F \supseteq \phi(X)$, donde F es el espacio l_2 de todas las secuencias

$$\psi = (\psi_1, \psi_2, ..., \psi_i, ...)$$

para las cuales

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

Esto inducirá un espacio definido por el vector de características como consecuencia de que una función lineal en F estará representada por

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

donde la primera expresión es la representación primal de la función y la segunda es la dual, la relación entre ambas representaciones viene dada por

$$\psi = \sum_{j=1}^{l} \alpha_j y_j \phi(x_j).$$

Observación 1. *El número de términos del sumatorio en la representación primal coincide con la dimensionalidad del espacio de características, mientras que en el dual tiene l términos (donde l es el tamaño de la muestra).*

Teorema 3 (Mercer)**.** *Sea X un subconjunto compacto de $\mathbb{R}^n$. Supongamos k una función continua y simétrica tal que el operados integral $T_k : L_2(X) \to L_2(X)$,*

$$(T_k f)(\cdot) = \int_X k(\cdot, x) f(x) dx,$$

es positivo, eso es

$$\int_{X\times X} k(x,z)f(x)f(z)dxdz \geq 0,$$

para toda $f \in L_2(X)$*. Entonces, podemos expandir k(x,z) en una serie uniformemente convergente (en* $X \times X$*) en términos de* $\phi_j \in L_2(X)$*, que son las autofunciones de los* $T_k's$*, normalizada de tal manera que* $\|\phi_j\|_{L_2} = 1$*, y los autovalores positivos asociados* $\lambda_j \geq 0$*,*

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

Las condiciones del teorema de Mercer son, por lo tanto, equivalentes a requerir que para cualquier subconjunto finito de X, la matriz correspondiente es semi-definida positiva. Las funciones que satisfacen esta propiedad las llamaremos kernel (o kernel de Mercer). Las características de Mercer proporcionan una representación de los puntos de entrada por medio de su imagen en el espacio de características con el producto interno definido por el número potencialmente infinito de autovalores del kernel. El subconjunto formado por la imagen del espacio de entrada está definido por los autovectores del operador kernel. Aunque las imágenes de los puntos $\phi(x) \in F$ no se computarán, sus productos internos se pueden calcular utilizando la función kernel.

2.3. Optimización de funciones

El objetivo final en los problemas de optimización es obtener los valores que maximizan o minimizan una función, pero considerando que dichos valores tienen que cumplir una serie de restricciones. La estructura general de este tipo de problemas es la siguiente:

$$\begin{array}{lll} \text{mín} & f(w), & w \in \Omega, \\ \text{s.a.} & g_i \leq 0, & i = 1, ..., k. \end{array}$$

Este tipo de problema, que se denomina *primal*, pretende obtener los valores de las *variables primales*, que están representadas por w, que minimizan la *función objetivo* $f(w)$. Las soluciones están sujetas a que tienen que respetar las *restricciones*, que pueden ser de igualdad, desigualdad o ambas en función del caso (en el anterior solo aparecen restricciones de desigualdad), $g_i(w)$. Si todas las funciones, f y las distintas g_i, son lineales se denomina *problema de programación lineal*; en cambio, si f es cuadrática y las restricciones lineales se dice *problema de programación cuadrática*. Al conjunto de todos los puntos del dominio que cumplen todas las restricciones se le llama *conjunto factible*, un elemento de dicho conjunto será el *óptimo*, que se denota por w^*. El óptimo va a ser el elemento del conjunto factible que verifica que para cualquier otro elemento del conjunto $f(w^*) \leq f(w)$.

2.3.1. Convexidad

La convexidad va a ser un punto clave en la resolución, ya que la convexidad del dominimo y de la función objetivo elimina la posibilidad de óptimos locales.

Definición 10. *Un dominio Ω es convexo si y solo si el segmento de la recta que une cualquier par de puntos del dominio está también incluido en el dominio,*

$$\Omega \text{ es convexo} \Leftrightarrow \forall u, v \in \Omega, \forall \theta \in (0,1) \text{ entonces } \theta u + (1-\theta)v \in \Omega.$$

En particular, en el caso de tener un problema cuadrático, si el dominio Ω es convexo entonces el conjunto factible también lo es, ya que las restricciones lineales no pueden eliminar la convexidad del dominio Ω.

Definición 11. *Una función $f : \mathbb{R}^d \to \mathbb{R}$ se dice que es convexa, si $\forall u, v \in \mathbb{R}^d$ y para cualquier $\theta \in (0,1)$,*

$$f(\theta v + (1-\theta)u) \leq \theta f(v) + (1-\theta) f(u).$$

Es decir, que el segmento de la recta que une el valor de la función en cualquier par de puntos, u y v, queda por encima del valor de la función en los puntos entre u y v. Si la desigualdad fuera estricta, entonces la función sería estrictamente convexa. En particular, una forma de ver que una función doblemente diferenciable

sea convexa es estudiar si su matriz Hessiana es semi-definida positiva.

Se dice que un problema es convexo cuando tanto el dominio, como la función objetivo y las restricciones son convexas. Este tipo de problemas son de gran interés gracias a la inexistencia de mínimos locales, lo que facilita la búsqueda del óptimo porque el primer mínimo encontrado va a ser también el mínimo global de la función.

Proposición 2. *Si una función f es convexa, entonces cualquier mínimo local w^* es también mínimo global.*

∗ **Demostración**:
Por definición de mínimo local, para cualquier $v \neq w^*$ existirá un θ suficientemente cerca de 1 tal que,

$$\begin{aligned} f(w^*) &\leq f(\theta w^* + (1-\theta)v) \\ f(w^*) &\leq (1-\theta)f(v) \\ (1-\theta)f(w^*) &\leq (1-\theta)f(v) \end{aligned}$$

luego $f(w^*) \leq f(v)$ para cualquier v, y por tanto w^* es mínimo global.

□

2.3.2. Dualidad

La teoría de Lagrange permite caracterizar las soluciones e indica cómo obtenerla. Fue establecida originalmente por Lagrange para problemas sin restricciones y completada por Kuhn-Tucker añadiendo el caso de restricciones de desigualdad. Los dos elementos principales de la teoría son los multiplicadores de Lagrange y la función lagrangiana.

Definición 12. *Dado el problema de optimización con función objetivo $f(w)$, definida sobre el dominio $\Omega \subseteq \mathbb{R}^d$ limitido por k restricciones $g_i(w) \leq 0$, se define la función lagrangiana como:*

$$L(w, \alpha) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w),$$

donde los distintos α_i se denominan multiplicadores de Lagrange y deben tener valores no negativos.

A los multiplicadores de Lagrange también se les denomina *variables duales*, que indica la importancia de cada restricción, a mayor valor, más difícil es cumplir la restricción. A partir de la función lagrangiana se puede definir el *problema dual* de la siguiente forma:

$$\begin{aligned} &\text{máx} \quad W(\alpha) = \inf_{w \in \Omega} L(w, \alpha), \\ &\text{s.a.} \quad \alpha_i \geq 0. \end{aligned}$$

El interés del problema dual es que puede ser más sencillo de resolver que el primal y, bajo ciertas condiciones, al resolverlo se puede obtener una solución del problema primal asociado.

Teorema 4. *(Dualidad débil)*

Sea $w \in \Omega$ una solución factible del problema primal y α una solución factible del problema dual, entonces $W(\alpha) \leq f(w)$.

∗ **Demostración**:

$$\begin{aligned} W(\alpha) = \inf_{v \in \Omega} L(v, \alpha) \leq \\ \leq L(w, \alpha) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) \leq \\ \leq f(w), \end{aligned}$$

Como w y α son factibles (del problema primal y dua respectivamente), entonces el sumatorio de las restricciones tiene que ser una cantidad negativa ya que las funciones $g_i(w)$ serán negativas y los α_i positivos.

□

Por tanto, de este teorema podemos sacar dos conclusiones:

1. El valor del problema dual está acotado superiormente por el primal:

$$\sup\{W(\alpha) : \alpha_i \geq 0\} \leq \text{ínf}\{f(w) : g(w) \leq 0\}$$

2. Si $f(w^*) = W(\alpha^*)$, respetándose las restricciones $g_i(w^*) \leq 0$ y $\alpha_i \geq 0$, entonces w^* y α^* son, respectivamente, las soluciones factibles del problema primal y dual.

A continuación vamos a ver el teorema de Kuhn-Tucker que indica las condiciones que deben cumplirse para poder resolver un problema primal gracias al dual.

Teorema 5. *(Teorema de Kuhn-Tucker)*

Sea un problema de optimización primal, donde tanto el dominio Ω como f son convexas y las restricciones g_i son funciones lineales, las condiciones necesarias y suficientes para que un punto w^* sea óptimo, es que exista α^* tal que

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

Al conjunto de todas estas condiciones se les denomina *condiciones de Karush-Kuhn-Tucker* o *condiciones KKT*. En caso de cumplirse, estamos garantizando que el valor en los óptimos del problema primal y dual coinciden ($f(w^*) = W(\alpha^*)$), ya que todos los sumandos añadidos a la función lagrangiana serían iguales a cero.

La segunda condición de las KKT, es decir, $\alpha_i^* g_i(w^*) = 0$ se denomina *condición complementaria*. Esta condición indica que para las restricciones g_i activas (aquellas que valen exactamente cero) su multiplicador de Lagrange puede ser mayor o igual que cero; sin embargo, para las inactivas (valen estrictamente menor que cero) el multiplicador asociado debe ser exactamente cero. Esto determina una propiedad muy importante de las SVM como es la dispersión de la solución.

En la práctica, partiendo del problema primal, se transforma en el dual igualando a cero las derivadas parciales de la función lagrangiana respecto de las variables

primales y sustituyendo las relaciones obtenidas de nuevo en la función lagrangiana.

De esta forma, la función dual va a contener como variables los multiplicadores de Lagrange y la tenemos que maximizar teniendo en cuenta que dichos multiplicadores deben ser no negativos. Si se cumplen todas las condiciones KKT, entonces al resolver el problema dual se tendrá resuelto también el primal.

Capítulo 3

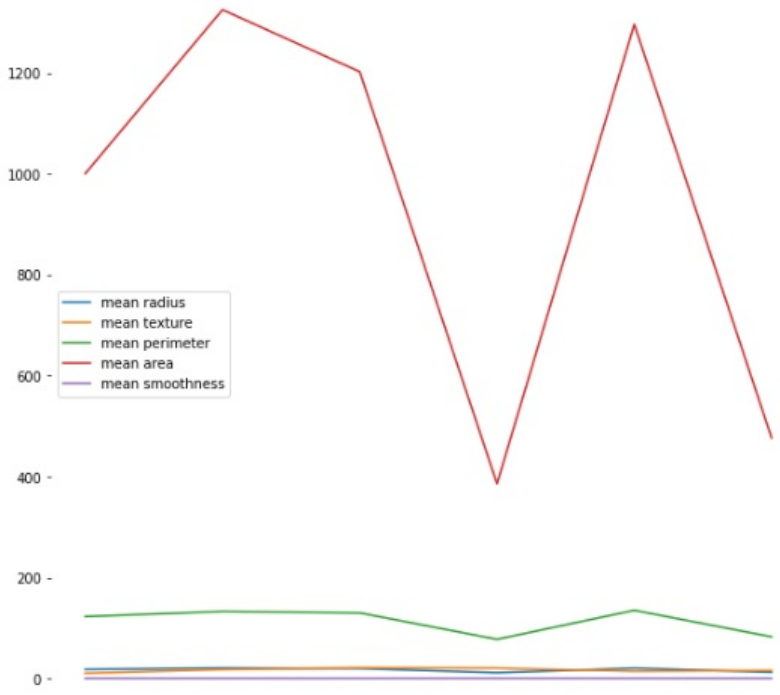
Experimentación

En este capítulo, se va a aplicar el desarrollo teórico que se ha visto en capítulos anteriores, de SVM y Métodos Kernel, a un caso concreto que podría ser real. Para ello, vamos a utilizar una base de datos estándar, pero el estudio no cambiaría si esa base hubiese sido de datos reales. El objetivo va a ser el mismo que se expuso al desarrollar la teoría, es decir, se parte de un conjunto de datos que van a estar separados en dos clases distintas, de forma que, se pretende construir el clasificador que separe las clases y, además, que sea el que mejor clasifique nuevos individuos.

La base de datos que se va a utilizar para realizar el estudio, es la base de datos de diagnóstico de cáncer de mama de Wisconsin. Esta base de datos está formada por 569 muestras y 32 atributos, que son reales, de las cuales una es el número ID y otra es el diagnóstico de si el tumor es benigno o maligno. Luego, en realidad, de los 32 atributos para el estudio no va a influir el ID. Para visualizar la forma que va a tener el conjunto de datos, se va a tomar, por ejemplo, las 5 primeras muestras (van a ser las filas) con los 5 primero atributos (que aparecen en columnas).

	mean radius	mean texture	mean perimeter	mean area	mean smoothness
0	17.99	10.38	122.80	1001.0	0.11840
1	20.57	17.77	132.90	1326.0	0.08474
2	19.69	21.25	130.00	1203.0	0.10960
3	11.42	20.38	77.58	386.1	0.14250
4	20.29	14.34	135.10	1297.0	0.10030
5	12.45	15.70	82.57	477.1	0.12780

Figura 3.1: Conjunto de datos de las 5 primeras muestras y los 5 primero atributos, sin contar con ID.



	0	1	2	3	4	5
mean radius	17.99	20.57	19.69	11.42	20.29	12.45
mean texture	10.38	17.77	21.25	20.38	14.34	15.7
mean perimeter	122.8	132.9	130.0	77.58	135.1	82.57
mean area	1001.0	1326.0	1203.0	386.1	1297.0	477.1
mean smoothness	0.1184	0.08474	0.1096	0.1425	0.1003	0.1278

Figura 3.2: Se visualiza el conjunto de datos en forma de gráfica. Tomando en este caso también las 5 primeras muestras y atributos.

Los atributos se va a separar en dos matrices, que llamaremos X e Y. Por un lado, la matriz X tiene los atributos quitando el número ID y el diagnóstico del tumor, es decir, es una matriz de 569 filas y 30 columnas. Por otro lado, Y que en realidad es un vector, de 569 filas, nos da las clases de las muestras, ya que es el atributo del diagnóstico del tumor. De forma que, si el tumor es benigno la variable Y de esa muestras concreta vale 1 y si es maligno vale 0. En particular, en dicha base de datos se tienen 212 muestras de tumores malignos y 357 de tumores benignos.

Sin embargo, para realizar el estudio, nos interesa dividir la base de datos en dos y de esta forma tener el conjunto de entrenamiento, a partir del cual se genera el clasificador, y el conjunto test, que van a ser los nuevos individuos que se pretenden clasificar correctamente mediante el clasificador. Esto se va a hacer de manera aleatoria utilizando el comando *train_test_split* y fijando una semilla para que siempre tome los mismos conjunto. De esta forma, se toma el 80 % de las muestras para el conjunto de entrenamiento y el 20 % para el conjunto test. En definitiva, el conjunto de entrenamiento, que se denota por (x, y), está formado por 455 de las 569 muestras, mientras que el conjunto test, (xtest, ytest), lo forman las 114 muestras restantes.

Con todo ello, se tiene lo necesario para comenzar la construcción del clasificador y el estudio del mejor clasificador para dicho conjunto de datos. Para realizar el estudio de los clasificadores se va a seguir siempre el mismo procedimiento: primero, entrenamos el clasificador con el conjunto de entrenamiento, seguidamente predecimos las y del conjunto de entrenamiento y, por último, predecimos las y del conjunto test calculando la bondad del modelo.

3.1. Kernel lineal

Como se mencionó anteriormente, en primer lugar, se va a construir el clasificador que entrenamos con el conjunto de entrenamiento (x, y).

```
from sklearn import svm
cl= svm.SVC(kernel='linear')
```

```
cl.fit(x,y)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Figura 3.3: Construcción del clasificador mediante SMV utilizando el kernel lineal, que seguidamente entrenamos con el conjunto x.

Seguidamente, se predicen las y del conjunto x, que denotaremos por y_pred. Se sabe que del conjunto de entrenamiento 291 muestras son tumores benignos y 164 malignos, en cambio la y_pred nos ha salido que tiene 298 tumores benignos y 157 malignos. Con lo que se deduce que va a haber muestras que el clasificador no predice correctamente su clase de pertenencia. Para ello, vamos a realizar un estudio para saber la bondad de ajuste del modelo.

```
import numpy as np
from sklearn.metrics import accuracy_score
accuracy_score(y, y_pred)
```

```
0.9626373626373627
```

Figura 3.4: Comando accuracy_score para ver bondad del modelo.

El comando *accuracy_score* calcula la precisión del clasificador para dicho conjunto. Este parámetro lo que indica es que el modelo ha acertado en un $96,26\,\%$ en la clasificación de las muestras del conjunto de entrenamiento. Asimismo, podemos visualizar la matriz de confusión (y la matriz de confusión normalizada), que nos da un recuento de los verdaderos benignos, falsos benignos, verdaderos malignos y falsos malignos, en nuestro caso.

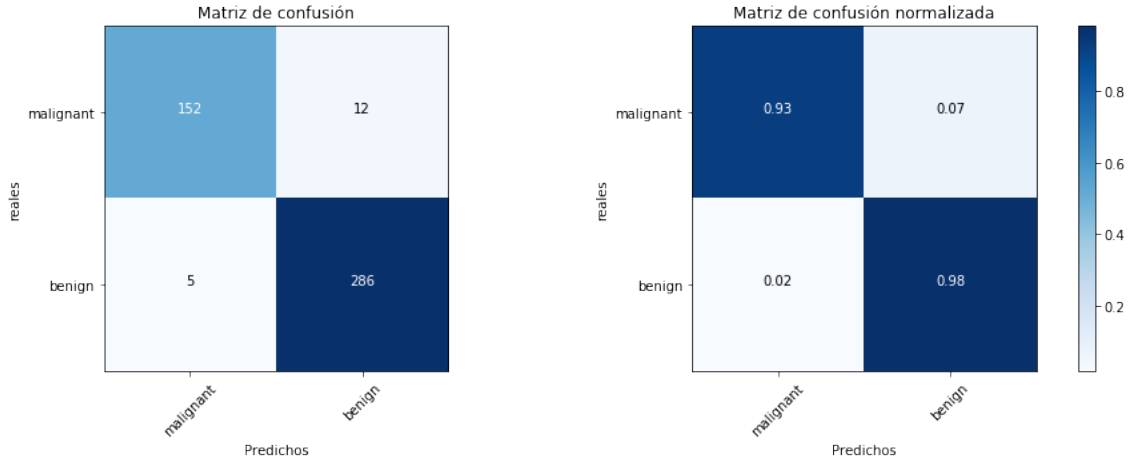


Figura 3.5: A la izquierda la matriz de confusión y a la derecha la matriz de confusión normalizada.

Gracias a la matriz de confusión se puede decir que de los 164 tumores malignos que hay en el conjunto de entrenamiento, 152 los ha predicho correctamente y los 12 restantes ha predicho que son tumores benignos (siendo malignos), mientras que de los 291 tumores benignos, 286 los ha clasificado correctamente y 5 los ha clasificado en la clase de los tumores malignos siendo benigno. Luego, el 93 % de los tumores malignos y el 98 % de los benignos han sido clasificados correctamente mediante el kernel lineal. Además, el error cuadrático medio sale un valor de 0.0373626.., que es un valor muy cercano a 0 con lo que podemos deducir que el clasificador utilizando un kernel lineal es bastante bueno para el conjunto de entrenamiento.

Se va a ver también una tabla que resume los resultados de bondad del modelo para cada clase. El comando *precision* calcula la presición, que es la habilidad del clasificador de no etiquetar como 0 una muestra que es de la clase 1 y viceversa (el mejor valor es 1 y el peor valor es 0). El comando *recall* calcula la memoria, que es la habilidad del clasificador para encontrar todas las muestras de una clase, (donde el mejor valor vuelve a ser 1 y el peor es 0). El $f1\_score$ calcula la puntuación f1, que se puede interpretar como un promedio ponderado de precisión y memoria, en el que la contribución relativa de ambas son iguales (alcanza su mejor valor en 1 y su peor valor en 0).

```
y_pred_class = np.int_(np.round(y_pred))
from sklearn.metrics import classification_report
print(classification_report(y,y_pred_class))
```

	precision	recall	f1-score	support
0.0	0.97	0.93	0.95	164
1.0	0.96	0.98	0.97	291
avg / total	0.96	0.96	0.96	455

Figura 3.6: Resumen de la precisión, memoria y puntuación f1 para cada clase.

Ahora, se va a realizar el mismo procedimiento pero para predecir los nuevos individuos del conjunto test (que son los que se buscan que estén correctamente clasificados), utilizando el clasificador que hemos entrenado con el conjunto de entrenamiento, que se va a denotar por ytest_pred. De esta manera, se obtiene que 67 de ellos se clasifican en la clase de los tumores benignos y 47 en la de los malignos. Mediante el comando *accuracy_score* se tiene que el modelo ha acertado un $97,37\,\%$ en la clasificación de los individuos del conjuntos test. En este caso, la matriz de confusión va a ser la siguiente,

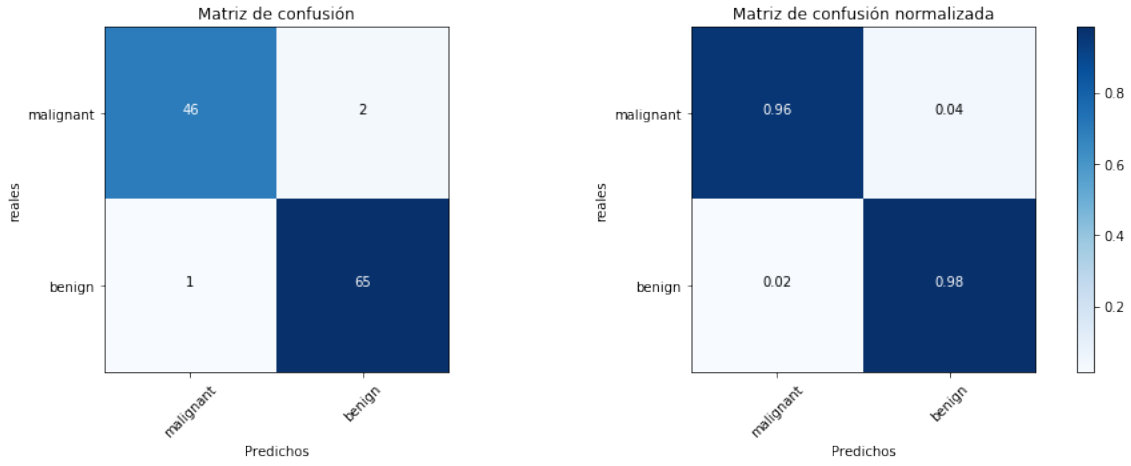


Figura 3.7: Matriz de confusión y matriz de confusión normalizada.

de la cual se tiene 46 y 65 de los tumores malignos y benignos están clasificados correctamente, es decir, el 96 % y 98 % respectivamente. Con un error cuadrático medio de 0.026315.., luego se puede concluir el estudio del clasificador utilizando un kernel lineal diciendo va a ser muy bueno para clasificar nuevos individuos en el conjunto de datos con el que se está trabajando.

```
ytest_pred_class = np.int_(np.round(ytest_pred))
print(classification_report(ytest,ytest_pred_class))
```

```
             precision    recall  f1-score   support

        0.0       0.98      0.96      0.97        48
        1.0       0.97      0.98      0.98        66

avg / total       0.97      0.97      0.97       114
```

Figura 3.8: Resumen de la precisión, memoria y puntuación f1 para cada clase del conjunto test.

3.2. Kernel gaussiano

En esta sección, se van a seguir los mismos pasos que se realizaron al hacer el estudio del clasificador utilizando un kernel lineal, pero en este caso se va a utilizar el kernel gaussiano. Por consiguiente, comenzamos construyendo el clasificador y entrenarlo con el conjunto x (de entrenamiento).

```
cr= svm.SVC(kernel='rbf')
cr.fit(x,y)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Figura 3.9: Construcción del clasificador mediante SMV utilizando el kernel RBF, que seguidamente entrenamos con el conjunto x.

Seguidamente, predecimos la y del conjunto de entrenamiento, y se obtiene que hay 291 muestras de la clase de los tumores benignos y 164 de los tumores malignos. Estos números coinciden con los que realmente hay por clase (como ya se nombró anteriormente), lo que no se sabe es si ha sido coincidencia o porque realmente ha clasificado correctamente el 100 % de las muestras. Para despejar esta incógnita, calculamos el parámetro *accuracy_score* que sale 1, lo que quiere decir que ha acertado el 100 %. Por tanto, este modelo va a ser a priori el modelo hipotético. De ahí que el error cuadrático medio sea 0 y la matriz de confusión tenga la siguiente forma,

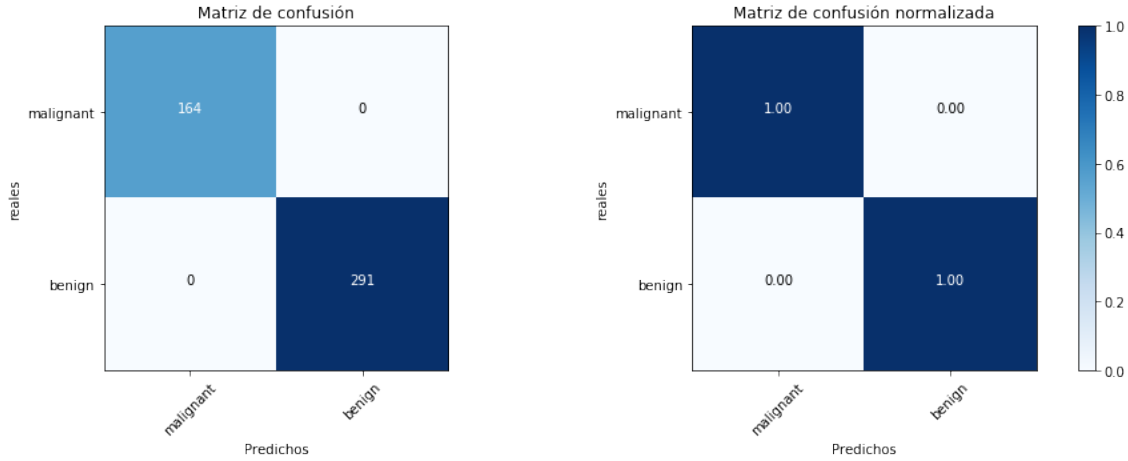


Figura 3.10: Matriz de confusión y matriz de confusión normalizada.

```
y_pred1_class = np.int_(np.round(y_pred1))
print(classification_report(y,y_pred1_class))
```

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	164
1.0	1.00	1.00	1.00	291
avg / total	1.00	1.00	1.00	455

Figura 3.11: Resumen de la precisión, memoria y puntuación f1 para cada clase.

Igualmente, se va a predecir los nuevos individuos del conjunto test utilizando dicho clasificador, que se va a denotar por ytest_pred1. De tal modo se obtiene que 114 se clasifican en la clase de los tumores benignos y ninguno en la clase de los malignos, con lo que acierta el $57,89\,\%$ y se tiene la matriz de confusión:

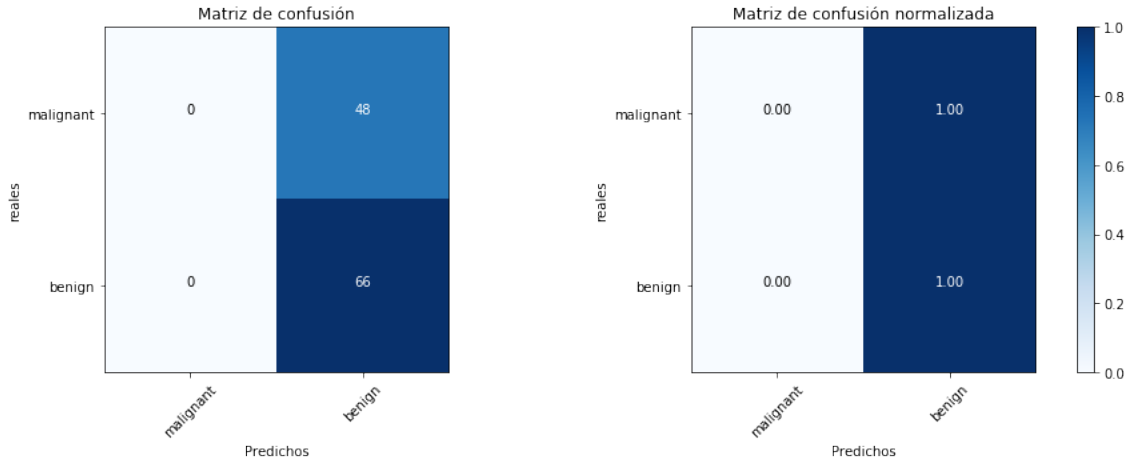


Figura 3.12: Matriz de confusión y matriz de confusión normalizada del conjunto test.

El error cuadrático medio es de 0,42105.., que ya no es un punto tan cercano a 0, lo que evidencia que este modelo no va a ser bueno a la hora de clasificar nuevos individuos.

```
ytest_pred1_class = np.int_(np.round(ytest_pred1))
print(classification_report(ytest,ytest_pred1_class))
```

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	48
1.0	0.58	1.00	0.73	66
avg / total	0.34	0.58	0.42	114

Figura 3.13: Resumen de la precisión, memoria y puntuación f1 para cada clase.

En definitiva, una vez realizado el estudio de ambos modelos, se pueden comparar los resultados con el objetivo de saber que modelo va a ser más eficiente para el conjunto de datos que se está estudiando. En primer lugar, se van a comparar mediante un histograma los resultados obtenidos para el conjunto de entrenamiento y, en segundo lugar, los obtenidos para el conjunto test. En dicho histograma los resultados utilizando el clasificador con kernel lineal van a aparecer en azul, mientras que los que utilizan el kernel gaussiano aparecen en verde.

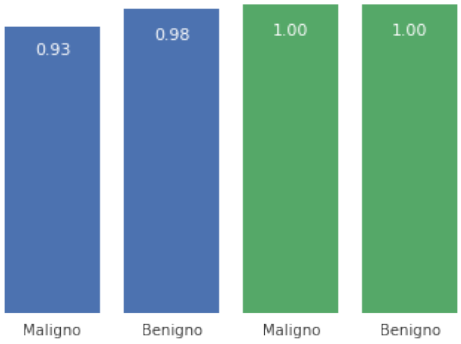


Figura 3.14: Comparación de las puntuaciones del conjunto de entrenamiento utilizando el clasificador linear frente a rbf (gaussiano).

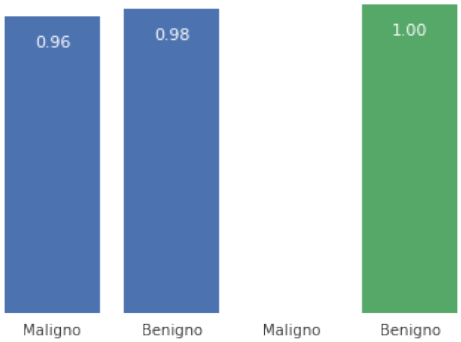


Figura 3.15: Comparación de las puntuaciones del conjunto test utilizando el clasificador linear frente a rbf (gaussiano).

Por consiguiente, cabe destacar que aunque aparentemente parece que el clasificador utilizando el kernel gaussiano va a ser el modelo más eficiente, se puede observar como no clasifica bien ninguna muestra que pertenece a la clase de los tumores malignos.

3.3. Conclusión

Como se mencionó anteriormente, la idea central no solo es buscar el clasificador que separe las clases sino que sea el que mejor clasifique nuevos individuos. En otras palabras, se busca que ante nuevos individuos, el clasificador sea un buen predictor. De esta manera, nos vamos a fijar en el error que comete un modelo sobre el conjunto test y no sobre el conjunto de entrenamiento, para ver la bondad del modelo. Podemos agrupar los datos sobre lo errores cuadráticos medios obtenidos en una tabla[1] de la siguiente forma,

Tabla de error cuadrático medio

	Conjunto de entrenamiento	Conjunto test
Modelo lineal	0,04	0,03
Modelo gaussiano	0	0,42

En realidad que haya un error bajo sobre el conjunto de entrenamiento es indicio de posible sobreajuste. Por tanto, para este caso concreto de conjunto de datos, el kernel gaussiano tiene un gran problema de sobreajuste ya que el hecho de que clasifique correctamente el 100 % de los datos del conjunto de entrenamiento no nos asegura que sea una buena herramienta predictiva (en tal caso, el error cuadrático medio es 0.42 que es bastante alto). El indicador que nos da la clave para saber cuál de los dos es mejor herramienta de clasificación es el error cuadrático medio sobre el conjunto test.

En contra de lo que cabía esperar, el modelo utilizando el clasificador lineal tiene un error cuadrático medio mucho menor que el modelo utilizando el clasificador

[1] En la tabla los errores están redondeados a dos decimales.

gaussiano. Por lo que podemos concluir que el mejor método para este problema de aprendizaje es el clasificador lineal.

Conviene puntualizar que estas conclusiones son relativas para el conjunto de entrenamiento concreto que estamos utilizando, sino los resultados cambiarían. En resumen, para estos datos de cáncer de mama, el modelo con clasificador lineal va a ser el más idóneo para predecir la clase de pertenencia de nuevos individuos.

Capítulo 4

Conclusiones

Con este trabajo de fin de grado se ha intentado dar evidencia de la gran importancia de los fundamentos matemáticos y su estudio para resolver problemas reales de la vida cotidiana y, en particular, de los problemas de clasificación que nos podemos encontrar. Desde problemas de clasificación de vinos, plantas, hasta problemas en el ámbito de la medicina, como es el caso de clasificación de mamografías.

De la misma forma que se ha realizado el estudio de separar la clase de los tumores benignos y malignos de una base estándar, se podría haber utilizado una base creada mediante pacientes concretos del hospital, ya que el estudio que se tendría que realizar seguiría los mismos pasos que se han realizado en el trabajo. Además, se podría haber tomado un caso en el que la clasificación no fuese binaria, en otras palabras, se podría haber tomado una base de datos que la clasificación fuera en 3 o más clases.

En conclusión, se podría seguir profundizando acerca de este tema utilizando otros métodos kernel apropiados para ciertos datos concretos, estudiando las diferentes propiedades matemáticas, los diferentes usos que podrían tener, etc.

Bibliografía

[1] V. N. Vapnik and S. Kotz, *Estimation of dependences based on empirical data*, vol. 40. Springer-Verlag New York, 1982.

[2] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Annals of statistics*, pp. 1651–1686, 1998.

[3] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," 1998.

[4] M. P. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares, and D. Haussler, "Knowledge-based analysis of microarray gene expression data by using support vector machines," *Proceedings of the National Academy of Sciences*, vol. 97, no. 1, pp. 262–267, 2000.

[5] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining (adaptive computation and machine learning)*. MIT press Cambridge, MA, 2001.

[6] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.

[7] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[8] I. Sebastian, "Fabrizio,"," *Machine Learning in Automated Text Categorization", ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[9] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[10] H. H. Zhang, J. Ahn, X. Lin, and C. Park, "Gene selection using support vector machines with non-convex penalty," *bioinformatics*, vol. 22, no. 1, pp. 88–95, 2005.

[11] E. Eskin, J. Weston, W. S. Noble, and C. S. Leslie, "Mismatch string kernels for svm protein classification," in *Advances in neural information processing systems*, pp. 1441–1448, 2003.

[12] J. R. Bock and D. A. Gough, "Predicting protein–protein interactions from primary structure," *Bioinformatics*, vol. 17, no. 5, pp. 455–460, 2001.

[13] S. Martin, D. Roe, and J.-L. Faulon, "Predicting protein–protein interactions using signature products," *Bioinformatics*, vol. 21, no. 2, pp. 218–226, 2004.

[14] W. A. Chaovalitwongse, Y.-J. Fan, and R. C. Sachdeo, "Novel optimization models for abnormal brain activity classification," *Operations Research*, vol. 56, no. 6, pp. 1450–1460, 2008.

[15] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[16] O. L. Mangasarian, W. N. Street, and W. H. Wolberg, "Breast cancer diagnosis and prognosis via linear programming," *Operations Research*, vol. 43, no. 4, pp. 570–577, 1995.

[17] D. Decoste and B. Schölkopf, "Training invariant support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 161–190, 2002.

[18] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Computer vision, 1998. sixth international conference on*, pp. 555–562, IEEE, 1998.

[19] A. Mucherino, P. Papajorgji, and P. M. Pardalos, "A survey of data mining techniques applied to agriculture," *Operational Research*, vol. 9, no. 2, pp. 121–140, 2009.

[20] N. Chen, *Support vector machine in chemistry*. World Scientific, 2004.

[21] J. Han, R. B. Altman, V. Kumar, H. Mannila, and D. Pregibon, "Emerging scientific applications in data mining," *Communications of the ACM*, vol. 45, no. 8, pp. 54–58, 2002.

[22] S. Lessmann and S. Voß, "Supervised classification for decision support in customer relationship management," in *Intelligent decision support*, pp. 231–253, Springer, 2008.

[23] G. Loveman, "Diamonds in the data mine," *Harvard business review*, vol. 81, no. 5, pp. 109–113, 2003.

[24] S. Meisel and D. Mattfeld, "Synergies of operations research and data mining," *European Journal of Operational Research*, vol. 206, no. 1, pp. 1–10, 2010.

[25] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen, "Using neural network rule extraction and decision tables for credit-risk evaluation," *Management science*, vol. 49, no. 3, pp. 312–329, 2003.

[26] J. H. Min and Y.-C. Lee, "Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters," *Expert systems with applications*, vol. 28, no. 4, pp. 603–614, 2005.

[27] T. Fawcett and F. Provost, "Adaptive fraud detection," *Data mining and knowledge discovery*, vol. 1, no. 3, pp. 291–316, 1997.

[28] G. Cui, M. L. Wong, and H.-K. Lui, "Machine learning for direct marketing response models: Bayesian networks with evolutionary programming," *Management Science*, vol. 52, no. 4, pp. 597–612, 2006.

[29] N. Glady, B. Baesens, and C. Croux, "Modeling churn using customer lifetime value," *European Journal of Operational Research*, vol. 197, no. 1, pp. 402–411, 2009.

[30] J. Hadden, A. Tiwari, R. Roy, and D. Ruta, "Computer assisted customer churn management: State-of-the-art and future trends," *Computers & Operations Research*, vol. 34, no. 10, pp. 2902–2917, 2007.

[31] Y.-L. Chen, K. Tang, R.-J. Shen, and Y.-H. Hu, "Market basket analysis in a multiple store environment," *Decision support systems*, vol. 40, no. 2, pp. 339–354, 2005.

[32] K.-W. Cheung, J. T. Kwok, M. H. Law, and K.-C. Tsui, "Mining customer product ratings for personalized marketing," *Decision Support Systems*, vol. 35, no. 2, pp. 231–243, 2003.

[33] R. Hendler and F. Hendler, "Revenue management in fabulous las vegas: Combining customer relationship management and revenue management to maximise profitability," *Journal of Revenue and Pricing Management*, vol. 3, no. 1, pp. 73–79, 2004.

[34] D. R. Morales and J. Wang, "Forecasting cancellation rates for services booking revenue management using data mining," *European Journal of Operational Research*, vol. 202, no. 2, pp. 554–562, 2010.

[35] P. L. Hammer, A. Kogan, and M. A. Lejeune, "Reverse-engineering country risk ratings: a combinatorial non-recursive model," *Annals of Operations Research*, vol. 188, no. 1, pp. 185–213, 2011.

[36] D. Bertsimas, M. V. Bjarnadóttir, M. A. Kane, J. C. Kryder, R. Pandey, S. Vempala, and G. Wang, "Algorithmic prediction of health-care costs," *Operations Research*, vol. 56, no. 6, pp. 1382–1392, 2008.

[37] V. V. Gavrishchaka and S. Banerjee, "Support vector machine as an efficient framework for stock market volatility forecasting," *Computational Management Science*, vol. 3, no. 2, pp. 147–160, 2006.

[38] K. P. Bennett and O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization methods and software*, vol. 1, no. 1, pp. 23–34, 1992.

[39] L. Xu, "Machine learning problems from optimization perspective," *Journal of Global Optimization*, vol. 47, no. 3, pp. 369–401, 2010.

[40] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, ACM, 1992.

[41] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[42] V. Vapnik, "The nature of statistical learning theory springer new york google scholar," 1995.

[43] V. Vapnik, *Statistical learning theory. 1998*. Wiley, New York, 1998.

[44] A. Palmer Pol and J. Y MONTAÑO MORENO, "¿ qué son las redes neuronales artificiales? aplicaciones realizadas en el ámbito de las adiciones," *Adicciones*, vol. 11, no. 3, pp. 243–255, 1999.

[45] B. Schölkopf, K. Tsuda, and J.-P. Vert, *Kernel methods in computational biology*. MIT press, 2004.

[46] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.