Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

MÁQUINAS DE VECTOR SOPORTE

Autor: Ángel Martín Marinas

Tutor: José Ignacio Farrán Martín

Año: 2023

Índice general

Capítulo 1

Introducción

Hoy en día gracias a la digitalización de los procesos, el número de datos que circulan por la red y son almacenados en las distintas bases de datos existentes es inmenso. En cualquier proceso que realicemos se almacenan un gran volumen de datos que en muchas ocasiones son útiles por diversos intereses. Sobre todos estos datos se pueden llevar a cabo distintos análisis con el fin de obtener una serie de resultados, por ejemplo, analizar los gustos cinematográficos de una cierta persona para poder sugerirla películas para visualizar, realizar predicciones médicas a cerca de una cierta enfermedad, clasificar especies de un cierto tipo de planta, clasificar un correo electrónico como no solicitado (spam) o solicitado.

Debido a este elevado volumen de datos, hace que el análisis de estos resulte una tarea lenta y costosa para los seres humanos, llegando a resultar inviable en muchos de ellos. Esta es la principal razón por la cual es necesario delegar esta actividad de análisis en las computadoras obteniendo un análisis automático de los datos y dando lugar al llamado Machine Learning. Cabe destacar que se delega en las computadoras tan solo la responsabilidad de los cálculos debido a su eficacia y rapidez en comparación con nosotros los humanos, siendo siempre los humanos los responsables de validar que los procedimientos empleados y los algoritmos que yacen por debajo están implementados correctamente y somos responsables de los resultados obtenidos por ellos. Por lo tanto, nosotros debemos de aportar el conocimiento matemático a cerca del problema que se nos plantee y formularlo correctamente, esta es la principal motivación de este trabajo.

Dentro del machine learning existen diferentes ramas de conocimiento en las que en cada una de ellas se tienen diferentes propósitos. En este trabajo nos vamos a centrar en lo que se conoce como aprendizaje supervisado, en concreto en la clasificación binaria a través de la máquina de vectores soporte. Se introducirán los fundamentos esenciales de esta teoría en los que resta de introducción y en parte del segundo capítulo de conceptos previos junto con el resto de conceptos matemáticos que nos van a ser necesarios a la hora de desarrollar la teoría subyacente.

En el tercer capítulo desarrollaremos en que consiste la clasificación lineal por medio de la máquina de vectores soporte para el caso de los problemas en los que los datos son linealmente separables, se explicará en que consiste la filosofía del método, y realizaremos los cálculos necesarios para implementar el algoritmo de clasificación.

En el cuarto capitulo, estudiaremos la clasificación lineal para los problemas que no son linealmente separables y para ello emplearemos la máquina de vectores soporte de margen suave. En donde se explicará el concepto de margen y como encontrar el margen óptimo.

Por último, en el quinto capítulo se estudiará la clasificación no lineal en problemas que no son linealmente separables, se explicará la filosofía que hay detrás de este método y se estudiarán las funciones empleadas para aplicar una clasificación no lineal, las cuales son denotadas como Kernels. Veremos las distintas propiedades que presentan dichas funciones, así como distintos métodos para su construcción.

La máquina de vectores soporte (MVS) tiene sus raíces en la teoría del aprendizaje estadístico y los métodos de optimización. Fueron introducidos por Vapnik en torno a 1990 y desde sus inicios han proporcionado técnicas efectivas para la minería de datos (data mining) y el aprendizaje automático (machine learning). Se han convertido en una herramienta muy poderosa debido a que son capaces de superar ciertas dificultades existentes en este tipo de técnicas como son la maldición de la dimensionalidad, el sobreajuste, etc. Otras de las cualidades destacables de las MVS son su cómoda representación matemática, así como las explicaciones geométricas que subyacen en todo momento, permiten una elevada generalización y su rendimiento es muy eficaz.

1.1. Descripción del problema

En esta sección se llevará a cabo una descripción del problema que se nos plantea y cual es la finalidad que buscamos aplicando las MVS.

El núcleo principal del que partimos es un conjunto de datos, generalmente en forma tabular o matricial, a la que se suele denominar una nube de datos. Cada uno de estos datos de forma general se corresponde con un individuo o una instancia de un determinado objeto de estudio, en el que se conoce una serie de propiedades a cerca de el. Para hacernos una primera imagen mental, podemos pensar que cada individuo u objeto a estudiar es una fila de la tabla y cada una de las columnas de dicha tabla se corresponde con un atributo o propiedad de dicho individuo.

El propósito general consiste en obtener patrones o ciertos comportamientos que se repitan a partir de los datos que se disponen. En concreto en la clasificación supervisada, se dispone de una serie de datos (individuos o lineas) con una serie de atributos a cerca de ellos y que pueden ser clasificados en distintas clases de acuerdo a algún criterio preestablecido. En el caso de este trabajo tan solo tendremos en cuenta dos posibles clases que asignar el individuo (clasificación binaria), debido a que si se considerarían más clases la complejidad del trabajo se excedería en la carga de créditos que este conlleva (12 créditos). Por ejemplo, supongamos que se dispone de unos cuantos individuos que presentan un cierto tumor y se dispone de distintos datos a cerca de dicho tumor, como puede ser las medidas, perímetro, textura, etc y queremos clasificar aquellos tumores en los que sean benignos y los que sean malignos en función a estos atributos.

Para ello, en primer lugar debemos partir de un conjunto de datos en los que se encuentran una serie de individuos previamente clasificados (clasificación supervisada), es decir, uno de los atributos de los individuos es la llamada etiqueta en la cual se indica el grupo al que pertenece el individuo en cuestión. A este conjunto de datos se le denomina el conjunto de entrenamiento y es a partir del cual vamos a obtener el conocimiento necesario para poder deducir ante un nuevo individuo a que grupo asignarle.

Para poder deducir la posible clase a asignar el individuo es necesario crear un modelo de clasificación a partir del conjunto de entrenamiento, que normalmente consiste en una función que se aplica sobre los parámetros del individuo y nos devuelve la clase que debería de corresponderse con el individuo, es decir, nos proporciona una estimación de la etiqueta que le corresponda.

Al proceso de búsqueda de dicho modelo que permita clasificar los datos de los cuales no disponemos que etiqueta deben de tener se le conoce como el proceso de entrenamiento. En este proceso tratamos de buscar e ir refinando un modelo que generalice de la mejor forma para aquellos datos de los que no conocemos su etiqueta.

En nuestro ejemplo a cerca de los tumores, disponemos de una serie de individuos con los atributos de su tumor y una etiqueta que indica el tipo de tumor, es decir benigno o maligno. A partir de estos individuos que corresponden el conjunto de entrenamiento se refina un cierto modelo de clasificación teniendo en cuenta los parámetros disponibles y la etiqueta que se le corresponde a cada uno, de esta forma se entrena el modelo obteniendo un modelo final. Una vez que disponemos de dicho modelo nos sirve como predictor para conocer cual debe ser la etiqueta asociada a un nuevo tumor con tan solo evaluar los parámetros del tumor sobre el predictor.

1.2. Ejemplo práctico con Python

En esta sección vamos a ilustrar con un pequeño ejemplo en código Python, el uso que tiene las máquinas de vectores soporte en el machine learning. Para ello, haremos una explicación a muy alto nivel de en que consiste cada una de las funciones que se emplean así como de los pasos que vamos dando. La idea es disponer de un esquema mental del propósito de uso para entender de una mejor forma el resto de la memoria. Para profundizar en los conocimientos acerca del uso de las máquinas de vectores soporte en Python, véase [1].

En este caso, vamos a coger un conjunto de datos de ejemplo que ya disponemos en una de las librería de Python. Los datos aquí utilizados son un conjunto de atributos acerca de tumores en una serie de individuos, tales como dimensiones, viscosidad, etc. Para cada uno de los individuos, se conocen los atributos del tumor y la etiqueta que lo determina como benigno o maligno. El objetivo es determinar ante un nuevo individuo con un tumor si este es maligno o benigno a partir de sus atributos. Los atributos vendrán dados en forma matricial en $\mathbf{X}$, donde cada fila es un individuo y cada columna es un atributo de dicho individuo. Las etiquetas de los distintos tumores de los individuos vienen dadas en un vector $\mathbf{y}$.

Una vez que disponemos de los datos, debemos de separarlos en el conjunto de entrenamiento y el conjunto de prueba, para ello utilizamos la función train_test_split indicando que el conjunto de test represente el 20 % del total.

```
from sklearn.model_selection import train_test_split

#Separo los datos de "train" en entrenamiento y
#prueba para probar los algoritmos

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
```

Después de esto, definimos el algoritmo, para ello importamos el algoritmo de máquina de vectores soporte ya definido en la librería sklearn por medio del comando sklearn.svm, y con esto ya podremos implementar el algoritmo en nuestro programa.

```
#Defino el algoritmo a utilizar
from sklearn.svm import SVC
algoritmo = SVC(kernel = 'linear')
```

Como vemos, hemos escogido un kernel lineal en nuestro algoritmo, el cual explicaremos más adelante en la memoria. Existen diferentes tipos de kernel a utilizar los cuales tan solo basta indicar cual queremos como parámetro.

En este punto, el paso que nos queda es entrenar el modelo, con el conjunto de entrenamiento.

```
#Entreno el modelo
algoritmo.fit(X_train, y_train)
```

Después de disponer del modelo entrenado, realizamos una predicción del conjunto de prueba o test.

```
#Realizo una prediccion
y_pred = algoritmo.predict(X_test)
```

Llegados a este punto, podemos comparar los resultados obtenidos por nuestro predictor, es decir, la clasificación en tumores benignos y malignos de los individuos del conjunto de test contra la clasificación real que conocemos de dicho conjunto de test, para obtener distintas métricas de nuestro modelo. Utilizaremos la matriz de confusión, que podemos importar del módulo sklearn.metrics .

El resultado obtenido se puede ver en la figura 1.1. Donde vemos que se han clasificado de forma correcta todos los individuos excepto 7 (los que no están en la diagonal de la matriz).

```
#Verifico la matriz de Confusion
from sklearn.metrics import confusion_matrix

matriz = confusion_matrix(y_test, y_pred)
```

```
Matriz de Confusión:
[[34  4]
 [ 3 73]]
```

Figura 1.1: Matriz de confusión

```
Precisión del modelo:
0.948051948051948
```

Figura 1.2: Precisión del modelo

```
print('Matriz de Confusion:')
print(matriz)
```

Por último, calculamos la precisión del modelo por medio de la función precision_score . La cual puede verse en la figura 1.2 indicando que el 94.8 % de los individuos se han clasificado de forma correcta.

```
#Calculo la precision del modelo
from sklearn.metrics import precision_score
precision = precision_score(y_test, y_pred)
print('Precision del modelo:')
print(precision)
```

Con esta información a cerca de nuestro modelo, se pueden ir variando distintos parámetros del modelo, como el kernel y visualizando como mejora o empeora nuestro modelo.

A continuación, se añade el código completo en Python que permite ejecutar el modelo.

```
"""
Maquinas Vectores de Soporte Clasificacion
"""
########## LIBRERIAS A UTILIZAR ##########

#Se importan la librerias a utilizar
from sklearn import datasets

########## PREPARAR LA DATA ##########

#Importamos los datos de la misma libreria de scikit-learn
dataset = datasets.load_breast_cancer()
print(dataset)

########## ENTENDIMIENTO DE LA DATA ##########

#Verifico la informacion contenida en el dataset
print('Informacion en el dataset:')
```

```
print(dataset.keys())
print()

#Verifico las caracteristicas del dataset
print('Caracteristicas del dataset:')
print(dataset.DESCR)

#Seleccionamos todas las columnas
X = dataset.data

#Defino los datos correspondientes a las etiquetas
y = dataset.target

####### IMPLEMENTACION DE MAQUINAS VECTORES DE SOPORTE #######

from sklearn.model_selection import train_test_split

#Separo los datos de "train" en entrenamiento
#y prueba para probar los algoritmos
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2)

#Defino el algoritmo a utilizar
from sklearn.svm import SVC
algoritmo = SVC(kernel = 'linear')

#Entreno el modelo
algoritmo.fit(X_train, y_train)

#Realizo una prediccion
y_pred = algoritmo.predict(X_test)

#Verifico la matriz de Confusion
from sklearn.metrics import confusion_matrix

matriz = confusion_matrix(y_test, y_pred)
print('Matriz de Confusion:')
print(matriz)

#Calculo la precision del modelo
from sklearn.metrics import precision_score
precision = precision_score(y_test, y_pred)
print('Precision del modelo:')
print(precision)
```

Capítulo 2

Conceptos previos

En este segundo capítulo de la memoria se introducirán algunos de los conceptos necesarios para desarrollar la teoría del resto de capítulos posteriores. El capítulo se dividirá en dos apartados. En un primer lugar se mostrarán ciertos resultados ya estudiados en el grado de matemáticas en forma de repaso, así como un desarrollo un poco más detallado de los conceptos a cerca de la optimización y la programación cuadrática. En segundo lugar, se mostrarán ciertos conceptos relacionados con el Machine Learning y el aprendizaje supervisado, los cuales nos van a servir para poder vincular nuestros propósitos con los conceptos matemáticos ya introducidos.

2.1. Conceptos matemáticos

En esta sección de la memoria, haremos un pequeño repaso de alguno de los resultados o definiciones ya vistos en el grado de matemáticas. Estos resultado y definiciones nos resultarán útiles para desarrollar el resto de capítulos. Para una perspectiva más amplia sobre las matemáticas que están detrás del Machine Learning, recomendamos [4].

Entre los conceptos que utilizaremos a lo largo del desarrollo de la memoria se encuentran conceptos de álgebra lineal, como son vectores, espacios vectoriales, matrices, aplicaciones lineales, aplicaciones afines y bases de espacios vectoriales. Por otra parte, también trataremos conceptos de geometría analítica, como son el producto interno,las normas,la proyección ortogonal y los ángulos. Y por último desarrollaremos con un poco más de detalle conceptos a cerca de optimización continua y programación cuadrática.

2.1.1. Álgebra lineal

Para no extender en exceso la memoria, nos limitaremos a comentar aquellas partes del álgebra lineal que nos serán necesarias para desarrollarla. Debido a que la mayoría de los conceptos algebraicos que debemos utilizar son bien conocidos por el lector en cuanto disponga de un mínimo conocimiento a cerca del álgebra lineal no vamos a centrarnos en las definiciones y teoremas que engloban esta parte del álgebra. Para obtener un desarrollo más completo de los resultados véase [3].

Dado que en todo momento vamos a estar manejando un gran volumen de datos, la estructura que debemos utilizar para tratar con estos datos serán los vectores y en consecuencia las agrupaciones de vectores dando lugar a las matrices. Suponemos conocida la forma de operar con matrices así como vectores por parte del lector, así como en que consiste la inversa y la traspuesta de una matriz. En consecuencia, al estar trabajando con vectores, nuestro marco de trabajo serán los espacios vectoriales, así como los subespacios vectoriales y las bases de vectores sobre los espacios vectoriales.

Otro de los conceptos algebraicos a tratar es el de las aplicaciones lineales y afines entre los distintos espacios vectoriales o afines. Lo que conlleva a la representación matricial de las aplicaciones lineales con respecto a las bases de los espacios vectoriales. También se tratarán las formas bilineales.

2.1.2. Geometría analítica

En esta sección daremos las definiciones y teoremas más importantes relacionados con la geometría analítica y que nos van a resultar útiles para el desarrollo del resto de la memoria. Para completar la información véase [3].

Dado a que como ya hemos comentado anteriormente vamos a tratar con vectores y espacios vectoriales, debemos de aportar una estructura geométrica los espacios vectoriales, proporcionar formas de medir vectores y distancias, así como medir el ángulo entre vectores. Para ser capaces de todo esto debemos de dotar a nuestro espacio vectorial de un producto interno. Para ello, daremos una serie de definiciones y teoremas relacionados con estos conceptos.

Definición 2.1. *Una norma en un espacio vectorial V, es una función*

$$\|\cdot\| : V \to \mathbb{R},$$
$$\mathbf{x} \mapsto \|\mathbf{x}\|$$

que asigna a cada vector x un número real, $\|\mathbf{x}\| \in \mathbb{R}$, de forma que para todo $\lambda \in \mathbb{R}$ y $\mathbf{x}, \mathbf{y} \in V$, cumple las siguientes propiedades:

- $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\|$
- *Desigualdad triangular:* $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$
- *Definida positiva:* $\|\mathbf{x}\| \geq \mathbf{0}$ *y* $\|\mathbf{x}\| = \mathbf{0} \Leftrightarrow \mathbf{x} = \mathbf{0}$

En nuestro caso, a lo largo de la memoria trabajaremos en todo momento con un ejemplo particular de norma, que es la norma euclídea, y trabajaremos en el espacio vectorial dado en $\mathbb{R}^n$,la cual se define de la siguiente forma.

Definición 2.2. *La norma euclídea de un vector $\mathbf{x} \in \mathbb{R}^n$ se define como*

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$$

La norma euclídea también es conocida como la norma l^2 y calcula la distancia del vector $\mathbf{x}$ al origen.

Definición 2.3. *Un producto interno sobre un espacio vectorial V es una forma bilineal, simétrica y definida positiva*

$$<>: V \times V \to \mathbb{C}$$

es decir que cumple las siguientes propiedades:

- *Simétrica:* $< \mathbf{x}, \mathbf{y} >= \overline{< \mathbf{y}, \mathbf{x} >}$
- *Bilineal:* $< \alpha\mathbf{x_1} + \beta\mathbf{x_2}, \mathbf{y} >= \alpha < \mathbf{x_1}, \mathbf{y} > +\beta < \mathbf{x_2}, \mathbf{y} >$ *(En la otra componente se deduce por la simetría)*
- *Definida positiva:* $\forall \mathbf{x} \in V \backslash \{\mathbf{0}\} :< \mathbf{x}, \mathbf{x} > > 0, < \mathbf{x}, \mathbf{x} >= 0 \iff \mathbf{x} = \mathbf{0}$

Al par $(V, < \cdot, \cdot >)$ se le llama espacio vectorial con producto interno.

En nuestro caso, utilizaremos el ejemplo concreto del producto interno que proporciona el producto escalar, y en caso de considerar el espacio vectorial $\mathbb{R}^n$ junto con el producto escalar como producto interno lo denominaremos espacio euclídeo.

Definición 2.4. *Definimos el caso especial de producto interno como producto escalar a la siguiente aplicación en $\mathbb{R}^n$*

$$\mathbf{x}^T\mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

Todo producto escalar induce una norma de la siguiente forma

$$\|\mathbf{x}\| := \sqrt{< \mathbf{x}, \mathbf{x} >}$$

pero sin embargo toda norma no viene inducida de un producto interno.

A partir de la norma, podemos definir la distancia.

Definición 2.5. *Consideramos el espacio vectorial con producto interno $(V, < \cdot, \cdot >)$, entonces*

$$d(\mathbf{x}, \mathbf{y}) := \|\mathbf{x} - \mathbf{y}\| = \sqrt{< \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} >}$$

se llama la distancia entre $\mathbf{x}$ e $\mathbf{y}, \forall \mathbf{x}, \mathbf{y} \in V$

En el caso de utilizar el producto escalar como producto interno, entonces a la distancia se la llama distancia euclídea.

Otro de los conceptos que nos proporciona el producto interno y la norma es el concepto de ángulo que se forma entre dos vectores del espacio vectorial.

Definición 2.6. *Se define el ángulo entre dos vectores $\mathbf{x}, \mathbf{y}$ como el único $\omega \in [0, \pi]$ con*

$$cos\omega = \frac{< \mathbf{x}, \mathbf{y} >}{\|\mathbf{x}\|\|\mathbf{y}\|}$$

Definición 2.7. *Dos vectores* $\mathbf{x}, \mathbf{y}$ *son ortogonales si y solo si* $< \mathbf{x}, \mathbf{y} > = 0$ *y escribimos* $\mathbf{x} \perp \mathbf{y}$. *Si además,* $\|\mathbf{x}\| = 1 = \|\mathbf{y}\|$ *son vectores unitarios, entonces son ortonormales.*

Una vez que hemos definido los conceptos a cerca de los ángulos entre los vectores, pasamos a definir en que consiste las proyecciones ortogonales sobre subespacios.

Definición 2.8. *Dado un espacio vectorial* V *y* $U \subseteq V$ *un subespacio vectorial de* V. *Una aplicación lineal* $\pi : V \to U$ *se llama proyección si* $\pi^2 = \pi \circ \pi = \pi$

Definición 2.9. *Consideremos un espacio vectorial con producto interno* $(V, < \cdot, \cdot >)$ *y en el un subespacio vectorial* U. *El conjunto de los vectores de* V *que son ortogonales a todos los de* U*:*

$$U^{\perp} = \{\mathbf{x} \in V | < x, u >= 0, \forall u \in U\}$$

es un subespacio de V *que es complementario de* $U (V = U \oplus U^{\perp})$ *y recibe el nombre de complemento ortogonal de* U.

Definición 2.10. *Consideremos un espacio vectorial con producto interno* $(V, < \cdot, \cdot >)$ *y en el un subespacio vectorial* U.*Puesto que* $V = U \oplus U^{\perp}$, *cada vector* $\mathbf{x} \in V$ *se expresa de forma única como suma de un vector de* U *y uno de* $U^{\perp}$*:*

$$\mathbf{x} = \mathbf{u} + \mathbf{w}; \mathbf{u} \in U, \mathbf{w} \in U^{\perp}$$

A la parte de U *se le llama a proyección ortogonal de* $\mathbf{x}$ *sobre* U *y se denota por* $u = p_U(x)$. $\mathbf{w}$ *es la proyección de* $\mathbf{x}$ *sobre* $U^{\perp}$. *La proyección ortogonal de* $\mathbf{x}$ *sobre* U *es, por tanto, el único vector* $\mathbf{u}$ *verificando simultáneamente que* $\mathbf{u} \in U$ *y que* $(\mathbf{x} - \mathbf{u}) \perp U$.

2.2. Conceptos a cerca de optimización y programación cuadrática

En esta sección de la memoria vamos a hacer una pequeña introducción sobre los conceptos que tienen que ver con la optimización de problemas de forma general y en concreto la resolución de problemas de programación cuadrática, los cuales van a ser utilizados en los apartados posteriores de la memoria. Nos centraremos en los resultados claves para el desarrollo de la memoria, para disponer un conocimiento más extendido, véanse [6] y [7].

En un primer lugar, mostraremos la forma general de un problema de optimización en un espacio N-dimensional.

$$\begin{cases} & \min f_0(\mathbf{x}), \mathbf{x} = (x_1, ..., x_N)^T \in \mathbb{R}^N \\ t.q. & \\ & f_i(\mathbf{x}) \leq 0, i = 1, ..., m \\ & h_i(\mathbf{x}) = 0, i = 1, ..., p \end{cases} \tag{2.1}$$

El vector $\mathbf{x}$ se llama la variable de optimización del problema, la función f_0 es la función objetivo. Las restricciones se corresponden con f_i y h_i, en concreto f_i restricciones de desigualdad y h_i las restricciones de igualdad. En el caso en el que $m + p = 0$ al problema

(2.1) se le conoce como problema sin restricciones, en el caso contrario es un problema con restricciones.

Definición 2.11. *(Punto factible y región factible) Un punto que satisfaga todas las restricciones del problema* (2.1) *se le llama punto factible. El conjunto de todos los puntos factibles del problema, constituyen la región factible* D*.*

$$D = \{\mathbf{x} | f_i(\mathbf{x}) \leq 0, i = 1, ..., m; h_i(\mathbf{x}) = 0, i = 1, ..., p; \mathbf{x} \in \mathbf{R}^N\}$$

Definición 2.12. *(Valor óptimo) El valor óptimo* p^* *del problema* (2.1) *se define como el ínfimo, es decir, la mayor cota inferior de la función objetivo* f_0 *en la región factible* D *cuando* D *es no vacío. En otro caso,* p^* *se define como infinito.*

$$p^* = \begin{cases} \text{ínf}\{f_0(\mathbf{x}) | \mathbf{x} \in D\} & si\ D \neq \emptyset \\ \infty & si\ D = \emptyset \end{cases}$$

Cabe destacar que el problema (2.1) es un problema de minimización, el hecho de que sea de minimización no supone ninguna restricción. De hecho, un problema de maximización puede ser reconvertido a uno de minimización con tan solo intercambiar el signo de la función objetivo f_0, así como en las restricciones del problema.

Repasemos en este punto las definiciones de conjunto convexo y función convexa.

Definición 2.13. *(Conjunto convexo) Un conjunto* $S \subset \mathbb{R}^N$ *es un conjunto convexo si el segmento que conecta a dos puntos cualesquiera del conjunto* S *esta contenido de forma completa en* S*, es decir, para cualesquiera* $\mathbf{x_1}, \mathbf{x_2} \in S$ *y cualquier* $\lambda \in [0, 1]$*, se tiene que*

$$\lambda \mathbf{x_1} + (1 - \lambda)\mathbf{x_2} \in S$$

Definición 2.14. *(Función convexa) Una función definida en* $\mathbb{R}^N$ *es una función convexa si para cualesquiera* $\mathbf{x_1}, \mathbf{x_2} \in \mathbb{R}^N$*, la gráfica de* f *está por debajo del segmento que une* $(\mathbf{x_1}, f(\mathbf{x_1}))$ *y* $(\mathbf{x_2}, f(\mathbf{x_2}))$*. Es decir, para cualquier* $\lambda \in [0, 1]$*,*

$$f(\lambda \mathbf{x_1} + (1 - \lambda)\mathbf{x_2}) \leq \lambda f(\mathbf{x_1}) + (1 - \lambda) f(\mathbf{x_2})$$

Cuando la desigualdad es estricta se conoce como función estrictamente convexa.

Teorema 2.1. *Se considera la función cuadrática en* $\mathbb{R}^N$

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H \mathbf{x} + \mathbf{r}^T \mathbf{x} + \delta$$

donde $H \in \mathbb{R}^{N \times N}, r \in \mathbb{R}^N, \delta \in \mathbb{R}$*. Si* H *es semidefinida positiva, entonces* $f(\mathbf{x})$ *es una función convexa en* $\mathbb{R}^N$*. Para el caso en el que* H *sea definida positiva será una función estrictamente convexa.*

Demostración. Se deduce haciendo uso de distintos teoremas y corolarios que se pueden ver en la primera sección de [7] y que no se han añadido por no extender este apartado. □

Daremos ahora la definición de problemas de programación convexa y cuadrática convexa.

Definición 2.15. *(Problema de programación convexa) Un problema de programación convexa es un problema de optimización con la siguiente forma:*

$$\begin{cases} & \text{mín } f_0(\mathbf{x}), \mathbf{x} = (x_1, ..., x_N)^T \in \mathbb{R}^N \\ t.q. & \\ & f_i(\mathbf{x}) \leq 0, i = 1, ..., m \\ & h_i(\mathbf{x}) = \mathbf{a_i}^T\mathbf{x} - b_i = 0, i = 1, ..., p \end{cases} \tag{2.2}$$

donde $f_0(\mathbf{x})$ y $f_i(\mathbf{x}), i = 1, ..., m$ son funciones continuas y convexas en $\mathbb{R}^N$ y $h_i(\mathbf{x}), i = 1, ..., p$ son funciones lineales.

Teorema 2.2. *Consideramos el problema de programación cuadrática siguiente:*

$$\begin{cases} & \text{mín } \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{r}^T\mathbf{x}, \mathbf{x} \in \mathbb{R}^N \\ t.q. & \\ & \bar{A}\mathbf{x} - \bar{b} \leq 0 \\ & A\mathbf{x} - b = 0 \end{cases}$$

donde $H \in \mathbb{R}^{N\times N}, \mathbf{r} \in \mathbb{R}^N, \bar{A} \in \mathbb{R}^{m\times N}, A \in \mathbb{R}^{p\times N}, \bar{b} \in \mathbb{R}^m, b \in \mathbb{R}^p$. Si H es semidefinida positiva, entonces el anterior problema es un problema de programación convexa, es decir, es un problema cuadrático de programación convexa.

Demostración. La prueba de que el problema es de programación convexa es inmediata usando el teorema 2.1 y la definición 2.15. □

A continuación, vamos a tratar la teoría dual que existe en los problemas de optimización, para ello veremos una serie de definiciones y resultados que se van a utilizar en el desarrollo de la memoria. Para disponer de conocimiento más en profundidad, véanse [6] y [7].

Consideremos el problema de optimización (2.2)

$$\begin{cases} & \text{mín } f_0(\mathbf{x}), \mathbf{x} = (x_1, ..., x_N)^T \in \mathbb{R}^N \\ t.q. & \\ & f_i(\mathbf{x}) \leq 0, i = 1, ..., m \\ & h_i(\mathbf{x}) = \mathbf{a_i}^T\mathbf{x} - b_i = 0, i = 1, ..., p \end{cases}$$

donde $f_i(\mathbf{x}), i = 0, 1, ..., m$ son funciones continuas, diferenciables y convexas en $\mathbb{R}^N$. Queremos estimar el valor óptimo p^* de la definición 2.12

$$p^* = \text{ínf}\{f_0(\mathbf{x})|\mathbf{x} \in D\}$$

donde D es la región factible del problema.

$$D = \{\mathbf{x}|f_i(\mathbf{x}) \leq 0, i = 1, ..., m; h_i(\mathbf{x}) = 0, i = 1, ..., p; \mathbf{x} \in \mathbf{R}^N\}$$

Para ello, introducimos la función Lagrangiana

$$L(\mathbf{x}, \lambda, \nu) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x}) \tag{2.3}$$

donde $\lambda = (\lambda_1, ..., \lambda_m)^T$ y $\nu = (\nu_1, ..., \nu_p)^T$ son los multiplicadores de Lagrange. Como es obvio, se tiene que cuando $\mathbf{x} \in D, \lambda \geq 0$, se tiene

$$L(\mathbf{x}, \lambda, \nu) \leq f_0(\mathbf{x})$$

Luego por tanto, tenemos la siguiente cadena de desigualdades

$$\inf_{\mathbf{x}\in\mathbb{R}^N} L(\mathbf{x}, \lambda, \nu) \leq \inf_{\mathbf{x}\in D} L(\mathbf{x}, \lambda, \nu) \leq \inf_{\mathbf{x}\in D} f_0(\mathbf{x}) = p^*$$

Luego, por tanto, introduciendo la función Lagrangiana dual:

$$g(\lambda, \nu) = \inf_{\mathbf{x}\in\mathbb{R}^N} L(\mathbf{x}, \lambda, \nu)$$

obtenemos

$$g(\lambda, \nu) \leq p^*$$

Por tanto, la desigualdad indica que para cualquier $\lambda \leq 0, g(\lambda, \nu)$ es una cota inferior de p^*. De todas estas cotas inferiores, aquella que sea la mayor nos proporciona conduce a la solución del problema de optimización.

$$\begin{cases} & \max g(\lambda, \nu) = \inf_{\mathbf{x}\in\mathbb{R}^N} L(\mathbf{x}, \lambda, \nu) \\ t.q. & \\ & \lambda \leq 0 \end{cases} \tag{2.4}$$

donde $L(\mathbf{x}, \lambda, \nu)$ es la función Lagrangiana de (2.3).

Definición 2.16. *(Problema dual) El problema* (2.4) *se llama problema dual del problema* (2.2)*. Y el problema* (2.2) *se llama problema primario o primal.*

Cabe destacar que si denotamos por d^* la solución del problema dual (2.4) entonces, se tiene:

$$p^* \leq d^*$$

Puesto que queremos que se cumpla la igualdad para disponer de la misma solución en ambos problemas, vamos a ver bajo que condiciones se cumple dicha igualdad. Dichas condiciones son las condiciones de Slater.

Definición 2.17. *(Condiciones de Slater) El problema de optimización primario* (2.2) *se dice que cumple las condiciones de Slater si existe un punto factible* $\mathbf{x}$ *de forma que*

$$f_i(\mathbf{x}) < 0, i = 1, ..., m; a_i^T\mathbf{x} - b_i = 0, i = 1, ..., p$$

O, cuando las primeras k restricciones de inecuación son restricciones lineales, existe un punto factible x de forma que

$$f_i(\mathbf{x}) = \bar{a_i}^T\mathbf{x} - \bar{b_i} \leq 0, i = 1, ..., k; f_i(\mathbf{x}) < 0, i = k+1, ..., m; a_i^T\mathbf{x} - b_i = 0, i = 1, ..., p$$

Teorema 2.3. *(Teorema de la dualidad fuerte) Consideremos el problema de programación primario* (2.2) *satisfaciendo las condiciones de Slater, y sea* p^* *el valor óptimo del problema primario* (2.2) *y* d^* *el valor óptimo del problema dual* (2.4)*, entonces:*

$$p^* = d^*$$

Demostración. No se incluye en la memoria debido a su extensión. Véase [2]. □

Definición 2.18. *(Condiciones KKT) Consideramos el problema de programación convexo* (2.2)*. Decimos que un punto* $\mathbf{x}^*$ *satisface las condiciones KKT si existen los multiplicadores* $\lambda^* = (\lambda_1^*, ..., \lambda_m^*)^T$ *y* $\nu^* = (\nu_1^*, ..., \nu_p^*)$ *de forma que la función Lagrangiana* (2.3) *satisface las siguientes condiciones:*

$$f_i(\mathbf{x}^*) \leq 0, i = 1, ..., m$$

$$h_i(\mathbf{x}^*) = 0, i = 1, ..., p$$

$$\lambda_{\mathbf{i}}^* \geq 0, i = 1, ..., m$$

$$\lambda_i^* f_i(\mathbf{x}^*) = 0, i = 1, ..., m$$

$$\nabla_x L(\mathbf{x}^*, \lambda^*, \nu^*) = \nabla f_0(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(\mathbf{x}^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\mathbf{x}^*) = 0$$

A continuación expondremos una serie de resultados sin demostración, para más detalles véanse [6] y [7].

Teorema 2.4. *Consideremos el problema de optimización primario* (2.2) *satisfaciendo las condiciones de Slater. Si* x^* *es su solución, entonces* x^* *satisface las condiciones KKT.*

Teorema 2.5. *Consideremos el problema de optimización primario* (2.2)*. Si* x^* *satisface las condiciones KKT, entonces* x^* *es su solución.*

Teorema 2.6. *Consideremos el problema de optimización primario* (2.2) *satisfaciendo las condiciones de Slater. Entonces, para su solución* x^* *es condición necesaria y suficiente que* x^* *satisfaga las condiciones KKT dadas en la definición 2.18.*

2.2.1. Resolución de problemas de programación cuadrática convexa

En esta sección vamos a tratar la forma general de resolución de problemas de programación cuadrática convexa. La resolución de este tipo de problemas va a ser necesaria en los posteriores capítulos de la memoria, en concreto en la obtención de los distintos algoritmos de clasificación.

Para ello, en un primer lugar, volvamos a recordar como están definidos los problemas de programación cuadrática convexa definición 2.15.

$$\begin{cases} & \text{mín } z = \mathbf{CX} + \mathbf{X^T DX} \\ t.q. & \\ & \mathbf{AX} \leq \mathbf{b} \\ & \mathbf{X} \geq \mathbf{0} \end{cases}$$

donde

$$\mathbf{X} = (x_1, ..., x_N)^T$$
$$\mathbf{C} = (c_1, ..., c_n)$$
$$\mathbf{b} = (b_1, ..., b_m)^T$$
$$\mathbf{A} = \begin{pmatrix} a_{11} & ... & a_{1N} \\ . & . & . \\ . & . & . \\ . & . & . \\ a_{m1} & ... & a_{mN} \end{pmatrix}$$
$$\mathbf{D} = \begin{pmatrix} d_{11} & ... & d_{1N} \\ . & . & . \\ . & . & . \\ . & . & . \\ d_{N1} & ... & d_{NN} \end{pmatrix}$$

La función $\mathbf{X^T DX}$ define una forma cuadrática. Asumimos que la matriz $\mathbf{D}$ es semidefinida positiva y simétrica, como hemos visto en el teorema 2.2 esto significa que z es convexa, puesto que las restricciones son lineales, se garantiza que el espacio de soluciones es convexo.

Haciendo uso de las condiciones KKT, definición 2.18, puesto que z es convexa y el espacio de soluciones es un conjunto convexo, entonces existe un punto óptimo global. Véase [6].

Al igual que los comentarios que se han hecho en apartados anteriores, el hecho de que el problema sea de minimización no es restrictivo, la conversión a un problema de maximización es directa, como ya se ha comentado.

Reescribiendo el problema en forma matricial, tenemos:

$$\begin{cases} & \text{mín } z = \mathbf{CX} + \mathbf{X^T DX} \\ t.q. & \\ & \mathbf{G(X)} = \begin{pmatrix} \mathbf{A} \\ -\mathbf{I} \end{pmatrix} \mathbf{X} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \leq \mathbf{0} \end{cases}$$

Fijando los multiplicadores de Lagrange correspondientes a las restricciones $\mathbf{AX} \leq \mathbf{b}$ y $-\mathbf{X} \leq \mathbf{0}$

$$\lambda = (\lambda_1, ..., \lambda_m)^T$$
$$\mathbf{U} = (\mu_1, ..., \mu_N)^T$$

Aplicando las condiciones KKT obtenemos

$$\lambda \geq \mathbf{0}, \mathbf{U} \leq 0$$

$$\nabla z - (\lambda^T, \mathbf{U}^T)\nabla \mathbf{G}(\mathbf{X}) = 0$$
$$\lambda_i(b_i - \sum_{j=1}^{m} a_{ij}x_j) = 0, i = 1, ..., m$$
$$\mu_j x_j = 0, j = 1, ..., n$$
$$\mathbf{AX} \leq \mathbf{b}$$
$$-\mathbf{X} \leq \mathbf{0}$$

Con esto, tenemos:

$$\nabla z = \mathbf{C} + \mathbf{2X^T D}$$
$$\nabla \mathbf{G}(\mathbf{X}) = \begin{pmatrix} \mathbf{A} \\ -\mathbf{I} \end{pmatrix}$$

Fijemos $\mathbf{S} = \mathbf{b} - \mathbf{AX} \geq \mathbf{0}$ las variables de holgura de las restricciones, entonces, las condiciones se reducen a:

$$-\mathbf{2X^T D} + \lambda^\mathbf{T}\mathbf{A} - \mathbf{U^T} = \mathbf{C}$$
$$\mathbf{AX} + \mathbf{S} = \mathbf{b}$$
$$\mu_j x_j = 0 = \lambda_i S_i, \forall i, j$$
$$\lambda, \mathbf{U}, \mathbf{X}, \mathbf{S} \geq \mathbf{0}$$

Puesto que $\mathbf{D}$ es simétrica, es decir $\mathbf{D^T} = \mathbf{D}$, la transpuesta de la primera de las ecuaciones queda de la siguiente forma:

$$-\mathbf{2DX} + \mathbf{A^T}\lambda - \mathbf{U} = \mathbf{C^T}$$

Por tanto, podemos combinar las condiciones necesarias de la siguiente forma:

$$\begin{pmatrix} -2\mathbf{D} & \mathbf{A^T} & -\mathbf{I} & \mathbf{0} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{X} \\ \lambda \\ \mathbf{U} \\ \mathbf{S} \end{pmatrix} = \begin{pmatrix} \mathbf{C^T} \\ \mathbf{b} \end{pmatrix}$$

$$\mu_j x_j = 0 = \lambda_i S_i, \forall i, j$$
$$\lambda, \mathbf{U}, \mathbf{X}, \mathbf{S} \geq \mathbf{0}$$

Excepto las condiciones $\mu_j x_j = 0 = \lambda_i S_i$, el resto de ecuaciones son funciones lineales en $\mathbf{X}, \lambda, \mathbf{U}$ y $\mathbf{S}$. Por tanto, el problema resultante es equivalente a resolver un conjunto de ecuaciones lineales con las condiciones adicionales $\mu_j x_j = 0 = \lambda_i S_i$.

La solución del sistema anterior se obtiene haciendo uso del método simplex, con su variante del método de la M. Teniendo en cuenta que la única restricción es que que satisfagan las condiciones $\mu_j x_j = 0 = \lambda_i S_i$. Estas significan que λ_i y S_i no pueden ser positivas de forma simultánea, así como tampoco pueden ser μ_j y x_j

Por último, puesto que z es convexa y el espacio de soluciones es convexo, la solución factible que cumpla todas las condiciones debe ser la única solución óptima.

2.3. Conceptos relacionados con el Machine Learning y el aprendizaje supervisado

En primer lugar, debemos tener en cuenta cuales son los tres pilares fundamentales del Machine Learning, los cuales son los datos, los modelos y el aprendizaje. Por ello, el primer paso que debemos de dar es estudiar cual es la forma de almacenar los datos así como hacer las operaciones necesarias sobre ellos.

En muchos de los problemas que nos encontramos los datos que debemos de analizar no se corresponden con datos numéricos sino categóricos, por lo cual este tipo de datos debemos de poder representarlos en un formato numérico, para ello ya existen numerosas estrategias, las cuales no vamos a desarrollar por no ampliar en exceso dicha memoria. Por lo cual en un principio, asumiremos que los datos que disponemos es posible representarlos en un formato numérico y que se encuentran en forma tabular. De esta forma, cada fila de la tabla representa un individuo o una instancia particular y cada columna de la tabla representa una característica o un atributo del individuo o instancia.

Otra de la problemática que nos podemos encontrar es que los datos no se ajusten a un formato tabular, como puede ser un texto o una imagen. En la actualidad existen numerosas técnicas que nos permiten obtener este tipo de datos en un formato tabular, las cuales no desarrollaremos en esta memoria por no extenderla.

Obviamente cuando hablamos de una tabla en el concepto de las matemáticas esto es equivalente a una matriz, así como las filas y las columnas se corresponden con los correspondientes vectores fila y columna de la matriz.

Otro aspecto importante a tener en cuenta a la hora de dar un formato correcto a los datos con los que vamos a trabajar es que las distintas columnas de la matriz de datos, es decir, las distintas características u atributos de los individuos pueden tener unas unidades totalmente distintas las unas a las otras, así como tratarse de magnitudes totalmente dispares. Para ello es importante reescalar dichos datos, es decir, para cada una de las columnas de nuestra matriz de datos (características de los individuos) debemos de normalizarla, es decir, hacer que su media sume 0 y su varianza sea 1.

Con todo esto, podemos asumir que partimos de una matriz de datos numéricos $\mathbf{X}$ de tamaño $N \times D$, en la que cada fila de la matriz X la denotamos por $\mathbf{x_n} \in \mathbb{R}^D, \forall n = 1, ..., N$, es decir, es un vector fila D-dimensional donde el índice D representa el número de características o atributos de un individuo o instancia y el índice N representa el número total de individuos o instancias que disponemos en nuestro conjuntos de datos.

Por otra parte, cada columna de la matriz X la representamos por $\mathbf{z_d} \in \mathbb{R}^N, \forall d = 1, ..., D$. Es decir se corresponde con un vector columna N-dimensional, que contiene el valor de una cierta característica d para cada uno de los N individuos.

En el problema de clasificación supervisada que tratamos en esta memoria, consiste en

clasificar cada uno de los individuos en un cierto grupo de acuerdo a ciertos criterios. En este caso tan solo vamos a considerar la clasificación binaria (clasificación en dos grupos) debido a su simplicidad y la complejidad de los cálculos en el caso de disponer de un mayor número de grupos en los que poder clasificar. Para ello, uno de los atributos que debemos de conocer de cada uno de los individuos del conjunto de datos que disponemos se corresponde con el grupo al que pertenece en la clasificación. Cabe destacar, que para el conjunto de datos que trabajamos el valor de este atributo es conocido, es decir, sabemos a que grupo pertenece cada uno de los individuos del conjunto de datos, esta es la justificación del adjetivo supervisada cuando hablamos de clasificación supervisada (conocemos el número posible de grupos a clasificar y la pertenencia de cada uno de los individuos del conjunto de datos a cada grupo).

Para este atributo (escalar) que indica el grupo al que pertenece un individuo, lo vamos a llamar etiqueta. Lo denotaremos de una forma especial $y_n, \forall n = 1, ..., N$. La cual indica a que grupo pertenece cada uno de los individuos $\mathbf{x_n}$.

De esta forma, podemos decir que un conjunto de datos se escribe como un conjunto de individuos-etiquetas o ejemplos-etiquetas de la siguiente forma.

$$\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_n}, y_n), ..., (\mathbf{x_N}, y_N)\}$$

Donde agrupando como hemos dicho anteriormente en forma matricial obtenemos la matriz $\mathbf{X} \in \mathbb{R}^{NxD}$.

Una vez que disponemos de un mecanismo de representación de los datos en forma de vectores y matrices podremos utilizar todos los conceptos que disponemos a cerca del álgebra lineal y la geometría aprendidos durante el grado de matemáticas y que han sido introducidos en el apartado anterior.

El segundo pilar fundamental del machine learning son los modelos o también llamados predictores. En la práctica hay dos enfoques distintos que se pueden utilizar para tratar de crear un modelo a partir de los datos. El primero de los enfoques trata de encontrar un modelo como una función que depende de los datos y el segundo de ellos es encontrar un predictor como un modelo probabilístico.

En el primero de los enfoques, podemos decir que un predictor es una función que dada una entrada particular produce una salida. En nuestro caso, la entrada de la función se corresponde con un vector $\mathbf{x_n} \in \mathbb{R}^D$ con las D características que se corresponde con el individuo que queremos clasificar. La salida que se correspondería con este predictor debería ser la clase a la que asignaríamos dicho individuo. Para modelar esto, podemos decir que la primera de la clase va a tomar el valor entero 1 y la segunda de las clases el valor entero -1. De esta forma podemos escribir el predictor como una función $f(\mathbf{x})$.

$$f : \mathbb{R}^D \to \{1, -1\}$$

A la hora de crear dichas funciones predictoras, podemos pensar en funciones lineales o funciones no lineales, lo cual se desarrollará en detalle en los próximos capítulos de la

memoria y se mostrarán sus ventajas e inconvenientes.

Para este punto de vista acerca de los predictores como funciones de los datos será necesario aplicar la teoría relativa al álgebra lineal y la geometría analítica vista en el capítulo anterior.

El otro enfoque consiste en considerar el predictor como un modelo probabilístico y no como una función única. De esta forma a partir de estos modelos probabilísticos podemos describir las funciones de distribución de la posible función predictora. En esta memoria no desarrollaremos este enfoque por no extendernos, por lo tanto consideraremos los predictores como funciones de los datos.

El tercer pilar fundamental es el aprendizaje o entrenamiento de los modelos. Una vez que se dispone de una forma de crear modelos o predictores, el siguiente objetivo consiste en obtener el mejor predictor, para ello debemos de cuantificar cuanto de bueno es un modelo y poder comparar cual de los dos predictores funciona mejor para un conjunto de datos dado, es decir, que predictor clasifica mejor para un conjunto de datos que a priori no conocemos a que clase pertenecen.

En este punto es donde entran en juego el conjunto de datos que disponemos $X \in \mathbb{R}^{N \times D}$. Este conjunto de datos recibe el nombre de conjunto de entrenamiento. A partir del conjunto de entrenamiento, podemos ajustar y entrenar los modelos (sus parámetros) de forma que cada vez sean capaces de predecir de una mejor forma a qué clase debe de asignarse un nuevo individuo cuando tan solo conocemos las características que lo describen. Para ello, es necesario definir una métrica para comparar los distintos modelos, seguiremos el principio de minimización del riesgo empírico, que básicamente proporciona un problema de optimización en el cual debemos de encontrar los parámetros de los modelos que mejor se ajusten.

Como hemos comentado desde el principio, el principal objetivo que buscamos es que nuestro modelo que ha sido entrenado a partir del conjunto de entrenamiento (conjunto de datos que disponemos) se ajuste de la mejor forma a aquellos datos nuevos que vamos a recibir y que deseamos clasificar, (y que no conocemos su etiqueta). Para simular el comportamiento de nuestro modelo predictor con aquellos datos que deseamos clasificar utilizaremos la validación cruzada.Todos estos conceptos se tratarán en los siguientes párrafos de la memoria.

En nuestro caso, en el cual consideramos los predictores como funciones que dependen de los datos del conjunto de entrenamiento, a la hora de entrenar los modelos nos basamos en la minimización del riesgo empírico, con la cual podremos estimar los parámetros que determinarán nuestro predictor.

Como hemos comentado nuestros predictores se basan en funciones, luego en un primer lugar debemos de establecer cual es la clase de las funciones que mejor se adaptan para nuestro objetivo. Para ello, supongamos que disponemos de un conjunto de datos de entrenamiento en forma matricial $\mathbf{X} \in \mathbb{R}^{NxD}$ con la notación comentada en los párrafos anteriores. Con estos datos de entrenamiento queremos estimar un predictor $f(., \theta) : \mathbb{R}^D \to \mathbb{R}$, que sea

parametrizado por θ. El propósito que buscamos es encontrar un parámetro θ^* que encaje bien para el conjunto de entrenamiento, es decir,

$$f(\mathbf{x_n}, \theta^*) \approx y_n, \forall n = 1, ..., N$$

Denotamos por $\hat{y_n} = f(\mathbf{x_n}, \theta^*)$, la salida obtenida por la función predictor para el individuo x_n con parámetro θ^*.

En muchos de los casos, el parámetro θ de los cuales dependen las clases de funciones que queremos considerar como predictores hace que esta clase de funciones se correspondan con funciones afines, dando posibilidades de que sean funciones afines lineales o no lineales como desarrollaremos a lo largo de la memoria.

Una vez que disponemos de la clase de funciones que nos interesa considerar como predictores y que están parametrizadas por el parámetro θ, debemos de encontrar cual de ellas es la que se considera el mejor predictor, es decir encontrar cual es el parámetro θ^* que mejor funciona como predictor. Para ello, debemos de tener en cuenta que el predictor se ajuste bien a los datos de entrenamiento, pero también muy importante que se ajuste bien a los nuevos datos de los nuevos individuos que queremos predecir. Para ello, debemos de en un primer lugar disponer de una forma de medir como de bien un predictor encaja con los datos que disponemos del conjunto de entrenamiento. Para ello definiremos lo que se considera como función de pérdida.

Puesto que partimos del conjunto de entrenamiento,$\mathbf{X} \in \mathbb{R}^{NxD}$, conocemos la etiqueta y_n que indica la clase a la que pertenece el individuo $\mathbf{x_n} \in \mathbb{R}^D$ y la correspondiente predicción $\hat{y_n}$ hecha a partir de $\mathbf{x_n}$ con la función predictor considerada. Con esto, definimos la función de pérdida como $l(y_n, \hat{y_n})$ y que produce como salida un número no negativo (al que se considera pérdida) que representa el error que se ha cometido con la predicción considerada. El objetivo que buscamos es encontrar un parámetro θ^* vectorial que minimice la pérdida o el error medio cometido en el conjunto de los N ejemplos que disponemos de entrenamiento.

Para ello, en un primer lugar, en machine learning es usual hacer una serie de suposiciones a cerca de los datos que disponemos del conjunto de entrenamiento; estas son que se corresponden con un conjunto de datos independientes e igualmente distribuidos. Esto conlleva que la media empírica sea un buen estimador de la media poblacional, y que podamos utilizar la media empírica de la pérdida sobre el conjunto de entrenamiento, para estimar la media de pérdida o error que cometemos.

Definimos lo que se conoce como riesgo empírico de la siguiente forma.

$$R_{emp}(f, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} l(y_n, \hat{y_n})$$

donde $\mathbf{X} \in \mathbb{R}^{N \times D}$ es la matriz del conjunto de entrenamiento, $\mathbf{y}$ el vector de etiquetas $\mathbf{y} := [y_1, ..., y_N]^T \in \mathbb{R}^N$, $\hat{y_n} = f(\mathbf{x_n}, \theta)$. El riesgo empírico depende de tres parámetros el predictor f, y los datos $\mathbf{X}, \mathbf{y}$.

Como ya hemos comentado, no nos interesan aquellos predictores que solo encajan o predicen bien para los datos del conjunto de entrenamiento; nuestro propósito es que predigan bien la clase de aquellos nuevos individuos de los que desconocemos cuál es su etiqueta (que tenga una pérdida pequeña) y que deseamos clasificar. De una manera formal, deseamos encontrar el predictor f (para ello debemos de fijar el parámetro θ) que minimice el riesgo esperado, el cual definimos de la siguiente forma.

$$R_{true}(f) = \mathbb{E}_{\mathbf{x},y}[l(y, f(\mathbf{x}))]$$

Donde y es la etiqueta, $f(\mathbf{x})$ es la predicción basada en un ejemplo $\mathbf{x}$ y $\mathbb{E}$ es la esperanza matemática. Como estamos tratando con la esperanza de una variable aleatoria el significado que tiene $R_{true}(f)$ es que este sería el valor del verdadero riesgo si tuviéramos acceso a un conjunto infinito de datos, los cuales tendrían cabida en el conjunto de todos los posibles datos y etiquetas $\mathbf{x}, \mathbf{y}$.

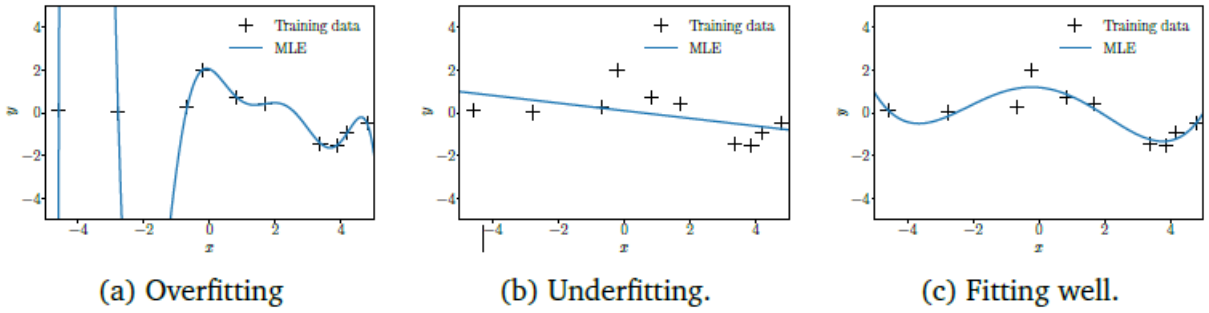
Cabe destacar que lo que denominamos el riesgo esperado también se le conoce como el riesgo poblacional, debido a que realmente estamos tratando de calcular la media poblacional de la función de pérdida.

El siguiente de los problemas que nos encontramos se corresponde con cómo conseguir que nuestro predictor generalice bien y sea capaz de predecir de forma correcta los nuevos individuos de los cuales no conoceremos su etiqueta, y por ende el grupo al que pertenecen. Para ello, debemos de simular aquellos nuevos individuos que vamos a recibir y que queremos clasificar. La forma de actuar consiste en dividir nuestro conjunto de datos $\mathbf{X} \in \mathbb{R}^{N \times D}$, del cual conocemos las etiquetas, y obtener dos porciones de datos, una de ellas la denominaremos el conjunto de entrenamiento, y a la otra el conjunto de test.

El conjunto de entrenamiento, al cual le denotaremos por el sufijo $_{train}$, le utilizaremos para entrenar y ajustar el modelo o predictor y el conjunto de test que le denotaremos por el sufijo $_{test}$ el cual no es tenido en cuenta por el algoritmo de machine learning durante el entrenamiento y ajuste de este, le utilizaremos para generalizar el rendimiento y evaluar como clasifica el predictor.

El objetivo que perseguimos es minimizar el riesgo empírico (y con ello el error en la clasificación) sobre el conjunto de datos del que conocemos las etiquetas. Esto puede producir que consigamos un riesgo empírico de 0, es decir, que nuestro predictor prediga de forma correcta todos los individuos de nuestro conjunto de datos; pero esto no tiene porque traducirse en que nuestro predictor vaya a clasificar de forma correcta los nuevos individuos que vamos a recibir y de los que no conocemos a que grupos pertenecen, es decir, nuestro predictor puede no generalizar bien para los nuevos datos. A este fenómeno se le conoce como sobreajuste.

Para intentar ver como se comporta nuestro predictor con los nuevos individuos de los que no sabemos su clasificación, es decir, ver como generaliza, utilizamos el conjunto de test, en el cual podemos comparar directamente el resultado de la etiqueta predicha por el predictor y el valor real de la etiqueta.



(a) Overfitting (b) Underfitting. (c) Fitting well.

Figura 2.1: Tipos de ajuste

El fenómeno del sobreajuste, en el cual se obtiene una media de pérdida muy pequeña sobre el conjunto de entrenamiento pero una pérdida muy grande sobre el conjunto de test, suele ocurrir cuando disponemos de un conjunto de datos muy pequeño y una clase de funciones predictoras que depende de parámetros bastante complejos.

Con la notación que hemos venido desarrollando, el sobreajuste ocurre cuando para un cierto predictor fijado (es decir, se fijan sus parámetros) el riesgo empírico estimado para el conjunto de entrenamiento $R_{emp}(f, \mathbf{X}_{train}, \mathbf{y}_{train})$ subestima el riesgo esperado $R_{true}(f)$, habiendo estimado el riesgo esperado $R_{true}(f)$ y usando el riesgo empírico del conjunto de test $R_{emp}(f, \mathbf{X}_{test}, \mathbf{y}_{test})$. Si el riesgo del conjunto de test es mucho mayor que el riesgo del conjunto de entrenamiento, es una señal de que es posible que exista el sobreajuste del modelo o predictor.

En general los modelos que tienen sobreajuste se suelen corresponder con modelos o predictores que tienen un elevado número de parámetros. Este elevado número de parámetros permite que el predictor sea muy flexible y se pueda adaptar a los datos que disponemos en el conjunto de entrenamiento, reduciendo el error de entrenamiento, lo cual provoca que cuando vayamos a utilizar el predictor para clasificar los nuevos individuos que desconocemos no funcione de forma correcta.

Por otra parte, otra posibilidad existente es que el modelo tenga un subajuste, que se corresponde con el problema opuesto al sobreajuste. En este caso, los modelos suelen tener un escaso número de parámetros y no son todo lo flexibles que deberían de ser, lo cual conduce que no se ajuste de la forma correcta a los datos de entrenamiento, provocando que el predictor no clasifique de la forma deseada al no reflejar la realidad.

El tercer caso, se corresponde con el caso deseado en el cual el modelo es correcto y se ajusta bien a los datos y no se produce ni sobreajuste ni subajuste. Esto indica que el predictor es lo suficientemente flexible para describir el conjunto de datos que disponemos.

En la figura 2.1 se puede visualizar de forma gráfica en que consiste cada uno de los casos.

Como hemos podido comprobar, no deseamos que nuestro predictor sufra de sobreajuste, para ello, debemos de alguna forma introducir algún término de penalización a la hora de minimizar el riesgo empírico que producen los estimadores, para encontrar el que mejor clasifica. A este término de penalización se le conoce como regularización. Con la regularización tratamos de establecer un compromiso entre la minimización del riesgo empírico y la complejidad del estimador que buscamos.

La forma de actuar consiste en incluir sobre la función de perdida $l(y_n, \hat{y_n})$ un parámetro de regularización λ sobre el propio regularizador, que suele ser del estilo de $\|\theta\|$, dado a que cuando los valores del vector de parámetros del predictor θ tiende a ser alejado del origen se suele incurrir en el sobreajuste. De esta forma se tendrá, que de forma general, la función de perdida será del siguiente estilo.

$$l(y_n, \hat{y_n}) = \{...\} + \lambda\|\theta\|$$

Donde $\{...\}$ dependerá de como se obtiene la función de pérdida dependiendo del problema en el que nos encontremos.

Una vez que hemos tratado con el hecho de cómo construir predictores que funcionen de forma correcta con aquellos nuevos individuos que deseamos clasificar y no conocemos sus etiquetas a partir del conjunto de entrenamiento que disponemos, debemos de tratar con la última cuestión, la cuál se corresponde a cual es el procedimiento para buscar el mejor predictor en el espacio de todos los posibles predictores que disponemos parametrizados.

Para poder comparar entre los distintos modelos o predictores existen diferentes técnicas, entre ellas la más utilizada es la llamada validación cruzada, que explicaremos a continuación. Como hemos comentado anteriormente, de nuestro conjunto de datos que disponemos con etiquetas vamos a hacer una partición, teniendo el conjunto de entrenamiento que destinaremos a entrenar el modelo y el conjunto de test o de validación, que destinaremos a evaluar como clasifica nuestro modelo. Como es lógico, nuestro deseo sería disponer de un conjunto de entrenamiento que sea lo suficientemente grande, al igual que un conjunto de test que sea también lo suficientemente grande, para que las estimaciones del rendimiento del predictor sean lo más fiables posibles (no tengan una varianza grande). Pero el problema que nos encontramos es que en la mayoría de los casos el conjunto de datos etiquetados que disponemos es bastante reducido.

Para lidiar con este problema y no sacrificar el tamaño de uno de los dos conjuntos (es decir, conjunto de entrenamiento grande para entrenar de forma correcta el modelo pero conjunto de test pequeño que puede no representar el rendimiento correcto del predictor, o conjunto de entrenamiento pequeño que hace que el modelo no se entrene bien y conjunto de test grande, que sí que proporciona mediciones de rendimiento reales pero el predictor no ha sido entrenado lo suficiente) se utiliza la validación cruzada, o también llamada validación cruzada de K hojas.

La validación cruzada divide el conjunto de datos etiquetados en K porciones, donde

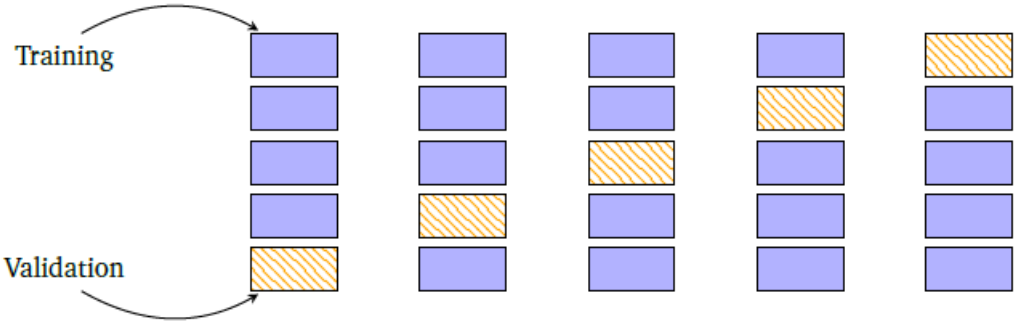


Figura 2.2: K-Hojas

$K-1$ de estas porciones se destinan al conjunto de entrenamiento del modelo R y la porción restante se destina al conjunto de test o validación V. De esta forma, con la validación cruzada tratamos de obtener todas las combinaciones posibles de las distintas asignaciones de las K porciones del conjunto de datos a los conjuntos R, V. En otras palabras, particionamos nuestro conjunto de datos D en dos conjuntos disjunto $D = R \cup V$ y $R \cap V = \emptyset$. De esta forma, entrenamos nuestro modelo sobre el conjunto R y después de haber sido entrenado, medimos el rendimiento del modelo entrenado sobre el conjunto de validación V.

De forma más precisa, para cada partición k, el conjunto de entrenamiento $R^{(k)}$ produce un modelo entrenado $f^{(k)}$, el cual es después aplicado y validado sobre el conjunto de test $V^{(k)}$, para calcular cual es el riesgo empírico $R_{emp}(f^{(k)}, V^{(k)})$. De esta forma repetimos el proceso sobre todas las posibles particiones a partir de las K hojas de los conjuntos de test y entrenamiento, y calculamos la media de los error de generalización del predictor, obteniendo el error de generalización esperado del predictor. Véase la figura 2.2

$$\mathbb{E}_V[R(f, V)] \approx \frac{1}{K} \sum_{k=1}^{K} R(f^{(k)}, V^{(k)})$$

Obviamente, una desventaja de la validación cruzada es que el coste computacional se aumenta, debido a que debemos de entrenar K veces el modelo, pero es una tarea que es fácilmente paralelizable y se puede entrenar cada uno de los K modelos de forma paralela, debido a que no dependen el uno del otro.

Como resumen, podemos decir que nuestro objetivo es encontrar aquel vector de parámetros θ^* que determine el modelo o predictor que mejor generaliza, como un problema de optimización en el que buscamos el mínimo del riesgo empírico, a partir de la función de perdida $R_{emp}(f, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} l(y_n, \hat{y_n})$, teniendo en cuenta que la función de pérdida $l(y_n, \hat{y_n})$ puede incluir regularización para forzar que el modelo generalice bien y no se cometa sobreajuste, es decir $l(y_n, \hat{y_n}) = \{...\} + \lambda\|\theta\|$. Una vez que dispongamos del modelo o predictor que mejor generalice, podemos estimar cual es su error de generalización esperado a través del método de la validación cruzada $\mathbb{E}_V[R(f, V)] \approx \frac{1}{K} \sum_{k=1}^{K} R(f^{(k)}, V^{(k)})$, pudiendo utilizarlo también para elegir el mejor modelo entre varios posibles candidatos.

Capítulo 3

Clasificación lineal con M.V.S. para problemas linealmente separables

Una vez que ya hemos introducido el trabajo que vamos a desarrollar en esta memoria y que disponemos del conocimiento previo necesario a cerca de los conceptos relacionados con el machine learning y el aprendizaje supervisado, y también hemos repasado algunos conceptos matemáticos necesarios para el desarrollo de la memoria, pasaremos a desarrollar el núcleo del trabajo.

Para ello nos centraremos en el algoritmo de aprendizaje supervisado para clasificación binaria de la máquina de vectores soporte (a lo largo de la memoria se abreviará por las siglas M.V.S.). En un primer lugar debemos de tener en cuenta que existen diversas variantes de este algoritmo y que vamos a ver a lo largo de la memoria; estas variantes son debidas a la naturaleza de los datos que disponemos.

Trataremos de ir de las variantes más sencillas a las más complejas. Para ello, en un primer lugar trataremos la clasificación lineal por medio de M.V.S. para problemas linealmente separables, en los cuales como deducimos de su nombre son utilizadas cuando los datos son linealmente separables, es decir que el conjunto de datos que queremos clasificar puede ser agrupado totalmente por medio de un separador lineal. Véase la figura 3.1.

En segundo lugar estudiaremos el caso de la clasificación lineal con M.V.S. para problemas no linealmente separables. En este caso, los datos pueden ser agrupados de forma parcial por medio de un separador lineal, pero existen puntos de la nube de datos que caen en el grupo incorrecto. Véase la figura 3.2

Por último, estudiaremos el caso de la clasificación no lineal en problemas no linealmente separables. En este caso, los datos no pueden ser agrupados por un separados lineal y debemos recurrir a los separadores no lineales. Véase la figura 3.3

En la sección actual vamos a entrar en detalle en la primera de las variantes. Para ello, en un primer lugar, como ya hemos comentado en los conceptos previos, partimos de un

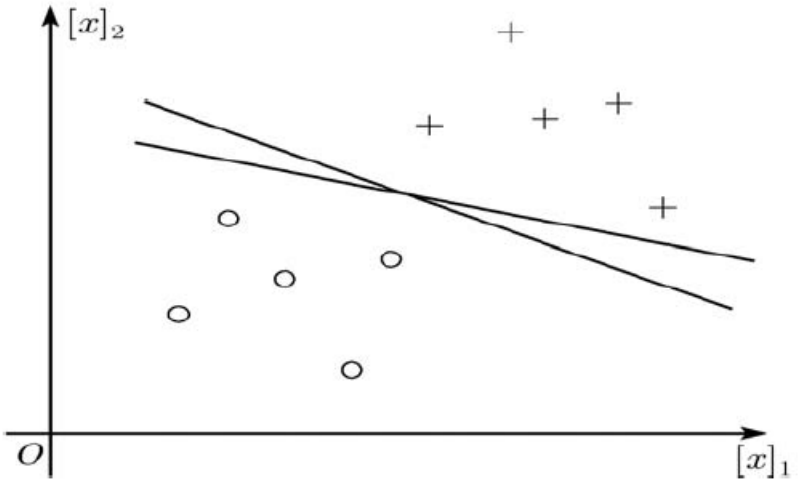


Figura 3.1: Linealmente separables

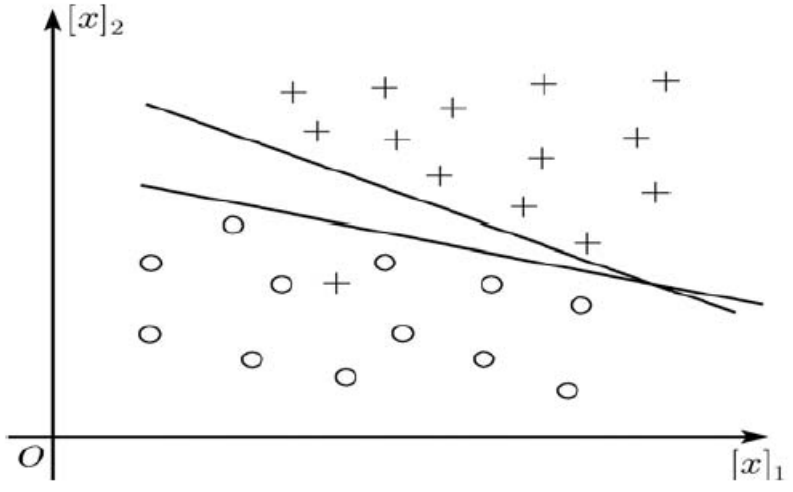


Figura 3.2: Parcialmente Linealmente separables

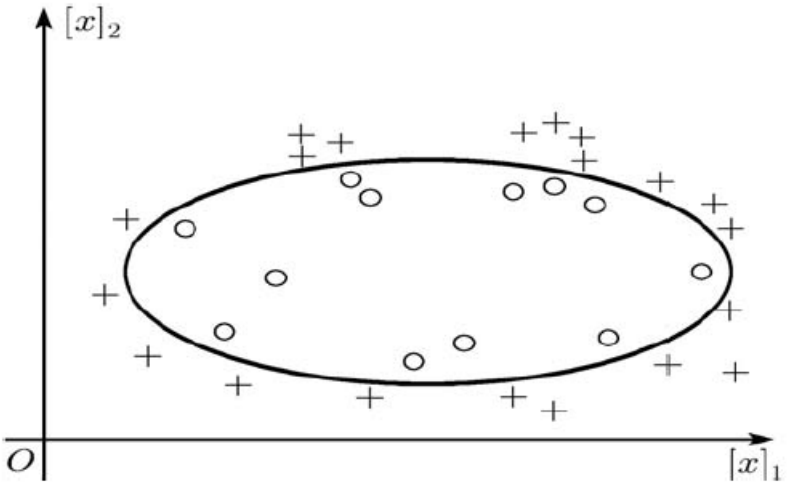


Figura 3.3: No linealmente separables

conjunto de entrenamiento de datos

$$\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_n}, y_n), ..., (\mathbf{x_N}, y_N)\}$$

donde agrupando en forma matricial obtenemos la matriz $\mathbf{X} \in \mathbb{R}^{N \times D}$. Nuestro propósito consiste en encontrar una función real $f(\mathbf{x})$ que actúe como predictor y nos proporcione el valor de $\hat{y_n} = f(\mathbf{x_n})$ para cada uno de los vectores de datos (individuos) $\mathbf{x_n}$, proporcionándonos la clase a la que dicho predictor asocia dicho individuo.

En resumidas cuentas, dado que estamos tratando con la clasificación binaria, existen dos posibles clases, por lo tanto tratamos de encontrar una forma de separar $\mathbb{R}^D$ en dos regiones de acuerdo al conjunto de entrenamiento $\mathbf{X}$. En el conjunto de entrenamiento, diremos que un punto $(\mathbf{x_i}, y_i)$ es un punto positivo del conjunto de entrenamiento si la clase a la que pertenece es la positiva, es decir, si su etiqueta es $y_i = 1$; para el caso en el que $y_i = -1$, diremos que es un punto negativo y pertenece a la clase negativa. Cabe destacar que cada $\mathbf{x_i}$ es un vector con D componentes.

En el caso en el que nos encontramos, los datos del conjunto de entrenamiento pueden ser agrupados o delimitados por medio de un separador lineal, es decir, los datos son linealmente separables. En este caso, forzamos a que la función predictora $f(\mathbf{x})$ se corresponda con una función lineal, es decir $f(\mathbf{x}) =< w, x > +b$ (donde denotamos con $<,>$ el producto escalar usual). En este caso el hiperplano $< w, x > +b = 0$ separa $\mathbb{R}^D$ en dos regiones. A continuación daremos la definición de un problema linealmente separable.

Definición 3.1. *(Problema linealmente separable). Consideremos un conjunto de entrenamiento* $\mathbf{X} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_n}, y_n), ..., (\mathbf{x_N}, y_N)\} \in (\mathbb{R}^D \times \Theta)^N$*,donde* $\mathbf{x_i} \in \mathbb{R}^D, y_i \in \Theta = \{1, -1\}, i = 1, ..., D$*. Si existe* $\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}$ *y un número positivo* ϵ *de forma que para todo subíndice i con* $y_i = 1$*, se tiene que* $< w, \mathbf{x_i} > +b \geq \epsilon$*, y para todo subíndice i con* $y_i = -1$*, tenemos que* $< w, \mathbf{x_i} > +b \leq -\epsilon$*. De esta forma decimos que el conjunto de entrenamiento y su correspondiente problema de clasificación son linealmente separables.*

Como hemos comentado, el hiperplano $< w, x > +b = 0$ actuará de separador de ambas clases. Dicho hiperplano se corresponde con un subespacio afín, el cual está parametrizado por el vector $w \in \mathbb{R}^D$ y $b \in \mathbb{R}$, donde w se corresponde con el vector normal al hiperplano.Para comprobarlo, tomemos $\mathbf{x_a}, \mathbf{x_b}$ en el hiperplano y veamos que el vector que une ambos es ortogonal a w.

$$f(\mathbf{x_a}) - f(\mathbf{x_b}) =< w, \mathbf{x_a} > +b- < w, \mathbf{x_b} > +b) =< w, \mathbf{x_a} - \mathbf{x_b} >$$

donde se ha utilizado la linealidad del producto escalar. Como hemos escogido $\mathbf{x_a}$ y $\mathbf{x_b}$ en el hiperplano, se cumple que $f(\mathbf{x_a}) = 0 = f(\mathbf{x_b})$ y por tanto $< w, \mathbf{x_a} - \mathbf{x_b} >= 0$. Dado que dos vectores son ortogonales cuando su producto interno es 0, hemos probado que w es el vector ortogonal al hiperplano.

Con todo esto, a la hora de clasificar un nuevo individuo (vector $\mathbf{x_i}$) calcularemos el valor de la función $f(\mathbf{x_i})$ y clasificaremos dicho individuo en en la clase positiva +1, si $f(\mathbf{x_i}) \geq 0$ y en clase negativa -1 en el caso contrario.

3.1. Máquina de vectores soporte primaria

Partiendo del caso en el que el conjunto de entrenamiento es linealmente separable, nos surge el primer problema que debemos de tratar a la hora de definir el hiperplano que hará de separador del conjunto de entrenamiento. Como es lógico, existen una infinidad de hiperplanos candidatos disponibles para escoger y como consecuencia clasificadores que son soluciones de nuestro problema de clasificación sin ningún error de entrenamiento.

Para tratar de escoger la mejor solución y tomar el hiperplano que mejor se adecua a nuestro problema de clasificación, la idea es escoger aquel hiperplano separador que maximice el margen entre los datos de entrenamiento que se corresponden con la clase positiva y la negativa, es decir, trataremos de tomar aquel hiperplano que haga que los datos de entrenamiento que corresponden a distintas clases sean separadas por el margen más amplio posible.

Para ello, trataremos con el concepto de margen. Daremos aquí una definición de margen.

Definición 3.2. *(Margen de un hiperplano). Es dos veces la distancia del hiperplano que actúa como separador a los ejemplos más cercanos del conjunto de datos, asumiendo que estamos en el caso en el que el conjunto de datos es separable.*

Una vez que conocemos la definición del margen, vamos a tratar de estudiar cual es la forma correcta de escoger el hiperplano separador de forma que sea lo más genérica posible a la hora de clasificar nuevos datos. Para ello, vamos a tratar de un ejemplo en dos dimensiones donde todo será mucho más visual al poder realizar dibujos. Para el caso de D dimensiones todo será igual, pero nos encontramos con la dificultad de no poder realizar dibujos que ilustren las ideas que se van a tratar.

En primer lugar, para estudiar de forma más sencilla la forma de proceder y estudiar en más detalle el concepto de margen, vamos a suponer que conocemos el vector normal

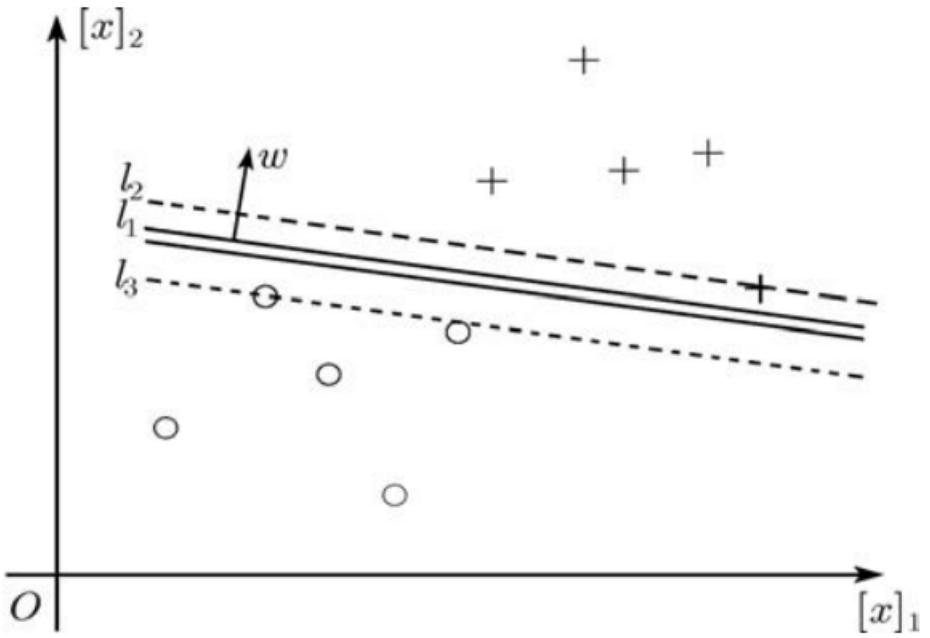


Figura 3.4: Margen de un hiperplano

w del hiperplano separador que tratamos de buscar. Al estar tratando con dimensión dos, dicho hiperplano se corresponde con una recta. Como podemos ver en la figura 3.4, la recta l_1 es una de todas las posibles rectas que separan de forma correcta el conjunto de datos de entrenamiento (los puntos positivos y negativos, donde cada uno representa una clase). Dicha recta no es única en este sentido, pues cualquier recta paralela a l_1 que cumpla con separar ambas clases nos sirve. Las rectas l_2 y l_3 de los extremos se las conoce como recta soporte. De todos las recta candidatas que existen entre ambas, la que se encuentra entre medias de ambas (en sentido de la que se encuentra a igual distancia de cada una de ellas), se corresponde con la mejor recta separadora de ambas clases y es la que tomaremos como hiperplano separador del problema. De esta forma, el margen es la distancia entre ambas rectas soporte.

Lo que acabamos de ver nos proporciona una forma de obtener el hiperplano separador óptimo una vez que conocemos la dirección del vector normal w, la cual no es conocida a priori en nuestro problema.

A continuación vamos estudiar como obtener el margen, para ello haremos un reescalado de los datos, de forma que el valor del predictor $< w, \mathbf{x} > +b$ sea uno en el ejemplo del conjunto de datos más cercano al hiperplano separador. Denotemos también a dicho punto más cercano como x_a, de forma que $< w, \mathbf{x_a} > +b = 1$. Denotemos por $\acute{x_a}$ la proyección ortogonal de x_a sobre el hiperplano separador, dado que $\acute{x_a}$ se encuentra en el hiperplano separador, tenemos que cumple la ecuación del hiperplano es decir

$$< w, \mathbf{\acute{x}_a} > +b = 0$$

Por otra parte, sabemos que x_a se encuentra a distancia del hiperplano r donde en este

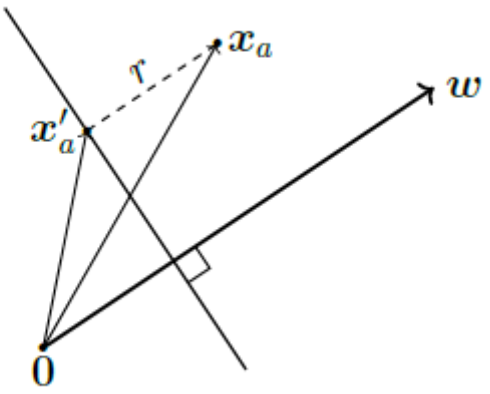


Figura 3.5: Margen de un hiperplano

caso r es la mitad del margen que queremos calcular, por haber escogido de esta forma x_a. Por tanto, podemos descomponer x_a de la siguiente forma a partir de las proyecciones ortogonales.

$$\mathbf{x_a} = \mathbf{x_a'} + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Con todo esto volviendo a la expresión $< w, \mathbf{x_a'} > +b = 0$ y sustituyendo x_a' obtenemos

$$< \mathbf{w}, \mathbf{x_a} - r\frac{\mathbf{w}}{\|\mathbf{w}\|} > +b = 0$$

Por la bilinealidad del producto escalar,

$$< \mathbf{w}, \mathbf{x_a} > +b - r\frac{< \mathbf{w}, \mathbf{w} >}{\|w\|} = 0$$

Puesto que por la suposición hecha, sabemos que $< \mathbf{w}, \mathbf{x_a} > +b = 1$, además $< \mathbf{w}, \mathbf{w} >= \|\mathbf{w}\|^2$, realizando las simplificaciones necesarios llegamos a que

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

Por tanto hemos obtenido el valor de la distancia r en términos del vector normal del hiperplano separador. El valor del margen se corresponde con $2r = \frac{2}{\|\mathbf{w}\|}$

Estudiemos ahora la forma en la que encontrar cual es la dirección del vector normal w que nos proporciona el separador óptimo, para ello trataremos de encontrar el hiperplano separador $< w, \mathbf{x} > +b = 0$ como un problema de optimización en las variables w y b.

Para ello, supongamos que el hiperplano separador es representado por $< \tilde{w}, \mathbf{x} > +\tilde{b} = 0$. Como ya hemos comentado, dicho hiperplano debe ser el hiperplano que se encuentre a la misma distancia de ambos hiperplanos soporte, por tanto ambos hiperplanos soporte pueden ser representados como $< \tilde{w}, \mathbf{x} > +\tilde{b} = k$ y $< \tilde{w}, \mathbf{x} > +\tilde{b} = -k$. Para tratar de agilizar la notación, consideremos que $w = \frac{\tilde{w}}{k}, b = \frac{\tilde{b}}{k}$, luego los hiperplanos soporte equivalentes se expresan como

$$< w, \mathbf{x} > +b = 1, < w, \mathbf{x} > +b = -1$$

Con esto, tenemos que el hiperplano separador es

$$< w, \mathbf{x} > +b = 0$$

puesto que conocemos que el margen se corresponde con $\frac{2}{\|\mathbf{w}\|}$, y nuestro propósito es que el hiperplano separador proporcione el mayor margen posible, esto deriva en un problema de optimización para w y b, que se formula de la siguiente forma:

$$\begin{cases} & \text{máx}_{w,b} \frac{2}{\|\mathbf{w}\|} \\ t.q. & \\ & < \mathbf{w}, \mathbf{x_i} > +b \geq 1, \forall i : y_i = 1 \\ & < \mathbf{w}, \mathbf{x_i} > +b \leq 1, \forall i : y_i = -1 \end{cases}$$

o de forma equivalente, podemos reformular el problema de la siguiente forma:

$$\begin{cases} & \text{mín}_{w,b} \frac{1}{2}\|\mathbf{w}\|^2 \\ t.q. & \\ & y_i(< \mathbf{w}, \mathbf{x_i} > +b) \geq 1, \forall i \end{cases}$$

De esta forma, podemos establecer el siguiente algoritmo, el cual indica la forma de proceder a la hora de calcular el hiperplano separador óptimo en el caso de problemas linealmente separables a partir del método de maximizar el margen.

Algoritmo 3.1. *(Método de maximización del margen para problemas linealmente separables).*

(1) Introducir el conjunto de entrenamiento $\mathbf{X} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_n}, y_n), ..., (\mathbf{x_N}, y_N)\} \in (\mathbb{R}^D \times \Theta)^N$*,donde* $\mathbf{x_i} \in \mathbb{R}^D, y_i \in \Theta = \{1, -1\}, i = 1, ..., D$.

(2)Construir y resolver el problema de optimización:

$$\begin{cases} & \text{mín}_{w,b} \frac{1}{2}\|\mathbf{w}\|^2 \\ t.q. & \\ & y_i(< \mathbf{w}, \mathbf{x_i} > +b) \geq 1, \forall i \end{cases} \tag{3.1}$$

obteniendo la solución $(\mathbf{w}^*, b^*)$.

(3) Construir el hiperplano separador $< \mathbf{w}^*, \mathbf{x} > +b^* = 0$ *y la función de decisión o estimador* $f(\mathbf{x}) = signo(< \mathbf{w}^*, \mathbf{x} > +b^*)$.

A este algoritmo en ocasiones también se le conoce como algoritmo de clasificación lineal de vectores soporte de margen duro, debido a las restricciones que supone el hecho de que requiere que todo el conjunto de entrenamiento debe ser clasificado completamente correctamente y no debe permanecer ninguno de los puntos del conjunto de entrenamiento en el "tubo" $-1 < (\mathbf{w} \cdot \mathbf{x}) + b < 1$. Más adelante veremos como relajamos estas restricciones y obtendremos el algoritmo de clasificación lineal de vectores soporte de margen suave.

3.2. Propiedades del método de margen máximo

Teorema 3.1. *Para un problema linealmente separable, existe solución* $(\mathbf{w}^*, b^*)$ *del problema de optimización y la solución, satisface:*
(i) $\mathbf{w}^* \neq 0$
(ii) Existe un $j \in \{i : y_i = 1\}$ *tal que*

$$< \mathbf{w}^*, \mathbf{x_j} > +b^* = 1$$

(iii) Existe un $k \in \{i : y_i = -1\}$ *tal que*

$$< \mathbf{w}^*, \mathbf{x_k} > +b^* = -1$$

Demostración. En primer lugar vamos a probar la existencia de la solución del problema, para ello, dado que nos encontramos en el caso de que el problema es linealmente separable, existe un punto factible $(\tilde{\mathbf{w}}, \tilde{b})$ del problema de optimización, por tanto con todo esto, el problema de optimización original es equivalente a :

$$\begin{cases} & \min_{w,b} \frac{1}{2}\|\mathbf{w}\|^2 \\ t.q. & \\ & y_i(< \mathbf{w}, \mathbf{x_i} > +b) \geq 1, \forall i \\ & \frac{1}{2}\|\mathbf{w}\|^2 \leq \frac{1}{2}\|\tilde{\mathbf{w}}\|^2 \end{cases}$$

Es sencillo ver que la región factible del problema anterior se corresponde con un conjunto cerrado, acotado y no vacío. Dado que el valor mínimo de una función continua $(\frac{1}{2}\|\mathbf{w}\|^2)$ se alcanza siempre en un conjunto cerrado, acotado y no vacío, concluimos con que existe solución del problema de optimización.

A continuación, vamos a probar las propiedades (i)-(iii):
(i) En este caso, para probar que $\mathbf{w}^* \neq 0$ basta con probar que $(\mathbf{w}^*, b^*) = (\mathbf{0}, b^*)$ no es solución. Para ello, procedemos por reducción al absurdo, supongamos que $(\mathbf{w}^*, b^*) = (\mathbf{0}, b^*)$ es solución, por ello, debería de satisfacer las restricciones del problema, por tanto

$$y_i(< \mathbf{0}, \mathbf{x_i} > +b^*) \geq 1, \forall i$$

es decir, $b^* \leq -1$ y $b^* \geq 1$, puesto que $< \mathbf{0}, \mathbf{x_i} >= 0, \forall i$ lo cual llegamos a una contradicción y hemos probado (i)
(ii) En este caso también por reducción a lo absurdo, supongamos que la conclusión no es cierta, es decir, que

$$< \mathbf{w}^*, \mathbf{x_i} > +b^* > 1, \forall i \in \{i, y_i = 1\} \tag{3.2}$$

Puesto que la solución $(\mathbf{w}^*, b^*)$ debe de satisfacer las restricciones, tenemos que:

$$< \mathbf{w}^*, \mathbf{x_i} > +b^* \leq 1, \forall i \in \{i, y_i = -1\} \tag{3.3}$$

Probemos que entonces $(\mathbf{w}^*, b^*)$ no es solución de (3.2) y (3.3) y llegaremos a una contradicción. Para ello, fijemos

$$\tilde{w} = \alpha w^*, \tilde{b} = (b^* + 1)\alpha - 1$$

Si $\alpha \in (0,1)$ entonces (3.3) es equivalente a

$$< \tilde{\mathbf{w}}, \mathbf{x_i} > +\tilde{b} \leq 1, \forall i \in \{i, y_i = -1\} \tag{3.4}$$

Por otra parte, para $i \in \{i, y_i = 1\}$ y de (3.2), tenemos que

$$lim_{\alpha \to 1^-}(< \tilde{\mathbf{w}}, \mathbf{x_i} > +\tilde{b}) = lim_{\alpha \to 1^-}[< \alpha \mathbf{w}^*, \mathbf{x_i} > +(b^* + 1)\alpha - 1] =< \mathbf{w}^*, \mathbf{x_i} > +b^* > 1$$

Por tanto, con esto tenemos que existe $\alpha \in (0,1)$ de forma que

$$< \tilde{\mathbf{w}}, \mathbf{x_i} > +\tilde{b} > 1, \forall i \in \{i, y_i = 1\} \tag{3.5}$$

Las desigualdades (3.4) y (3.5) indican que $(\tilde{\mathbf{w}}, \tilde{b})$ es un punto factible del problema de optimización y el valor que tiene en dicho punto la función objetivo es $\frac{1}{2}\|\tilde{\mathbf{w}}\| = \alpha^2 \frac{1}{2}\|\mathbf{w}^*\| < \frac{1}{2}\|\mathbf{w}^*\|^2$, lo que implica que $(\mathbf{w}^*, b^*)$ no es solución. Esta contradicción prueba la conclusión. (iii) Se prueba de forma similar a (ii). □

Cabe destacar que las conclusiones que nos proporciona este teorema es que de (i) el algoritmo de método de maximación de margen para problemas linealmente separables permite construir siempre un hiperplano separador de ambas clases, así como que de (ii) y (iii) indica que los dos hiperplanos $< \mathbf{w}^*, \mathbf{x} > +b) = \pm 1$ obtenidos en dicho algoritmo son los dos hiperplanos soporte.

A continuación, veremos un teorema que nos asegura la unicidad del hiperplano separador construido con el algoritmo.

Teorema 3.2. *Para un problema linealmente separable, la solución del problema de optimización es única.*

Demostración. Supongamos que existen dos soluciones del problema de optimización, sean estas $(\mathbf{w_1^*}, b_1^*)$ y $(\mathbf{w_2^*}, b_2^*)$, se puede probar que la solución del problema con respecto a la variable $\mathbf{w}$ es única, esto es que $\mathbf{w_1^*} = \mathbf{w_2^*}$. Esto es debido a que la solución de un problema de programación convexa con respecto a una de las variables es siempre única cuando la función objetivo es una función convexa, para más detalle véase [7] . Por tanto, las dos soluciones pueden ser reescritas como $(\mathbf{w}^*, b_1^*)$ y $(\mathbf{w}^*, b_2^*)$. De la conclusión (ii) del teorema 3.1, podemos decir que existen $j, j' \in \{1, ..., N\}$ tales que $y_j = y_{j'} = 1$ y

$$< \mathbf{w}^*, x_j > +b_1^* = 1$$

$$< \mathbf{w}^*, x_{j'} > +b_1^* \geq 1$$

$$< \mathbf{w}^*, x_{j'} > +b_2^* = 1$$

$$< \mathbf{w}^*, x_j > +b_2^* \geq 1$$

De donde deducimos que $b_1^* \geq b_2^*$ y $b_2^* \geq b_1^*$, luego por tanto concluimos que $b_1^* = b_2^*$, y en consecuencia $(\mathbf{w_1^*}, b_1^*) = (\mathbf{w_2^*}, b_2^*)$ y la solución del problema es única. □

3.3. Máquina de vectores soporte dual

La descripción de la máquina de vectores soporte de la sección anterior en términos de $\mathbf{w}$ y b, se la conoce como máquina de vectores soporte primaria o primal. Recordemos que en ese caso, estamos considerando las entradas, es decir los datos $\mathbf{x} \in \mathbb{R}^D$, con D características o propiedades sobre cada uno de los individuos. Por tanto, esto hace que $\mathbf{w}$ sea de la misma dimensión que $\mathbf{x}$, lo cual conlleva que el número de parámetros (la dimensión de $\mathbf{w}$) del problema de optimización aumente de forma lineal con el número de características o propiedades de los individuos, donde normalmente este suele ser un número elevado. Es decir, el número de parámetros va a crecer con respecto a D

En esta sección vamos a considerar un problema de optimización equivalente (su dual), el cual va a ser independiente del número de características de los individuos, pero sin embargo el número de parámetros del problema de optimización va a crecer con el número de ejemplos que vamos a tener en el conjunto de datos, es decir va a crecer con respecto a N.

Cabe destacar que utilizar un algoritmo u otro va a depender de la naturaleza de los datos del conjunto de datos de entrenamiento, en el caso en el que el conjunto de datos tenga menos número de ejemplos que el número de características que se almacenan de cada dato, es decir $(D > N)$ es mejor utilizar la máquina de vectores soporte dual, sin embargo, para los conjuntos de datos en los que se disponga un mayor número de ejemplos que el número de características que se almacena de cada individuo $(D < N)$ es mejor utilizar la máquina de vectores soporte primaria.

Para formular el problema de optimización equivalente, haremos uso del problema dual de optimización, para ello es necesario obtener la función Lagrangiana del problema de optimización utilizado en la máquina de soporte primaria. Dicha función lagrangiana, se corresponde con:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}^2\| - \sum_{i=1}^{l} \alpha_i (y_i(< \mathbf{w}, \mathbf{x_i} > +b) - 1) \tag{3.6}$$

donde $\alpha = (\alpha_1, ..., \alpha_l)^T$ es el vector de los multiplicadores de Lagrange.

A continuación se expondrán algunos de los teoremas necesarios para obtener la solución del problema de optimización dual:

Teorema 3.3. *El problema de optimización:*

$$\begin{cases} & \text{máx}_\alpha \, -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j < x_i, x_j > \alpha_i \alpha_j + \sum_{j=1}^{l} \alpha_j \\ t.q. & \\ & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & \alpha_i \geq 0, i = 1, ..., l \end{cases} \tag{3.7}$$

Es el problema dual del problema de optimización primario (3.1).

Demostración. Teniendo en cuenta la definición del problema dual de optimización, definición 2.16, el problema dual aquí considerado, sería el siguiente:

$$\begin{cases} & \max g(\alpha) = \inf_{\mathbf{w},b} L(\mathbf{w}, b, \alpha) \\ t.q. & \\ & \alpha \geq 0 \end{cases}$$

Como $L(\mathbf{w}, b, \alpha)$ es una función cuadrática estrictamente convexa con respecto a la variable $\mathbf{w}$, su valor mínimo se alcanza para $\mathbf{w}$ satisfaciendo:

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \alpha) = \mathbf{w} - \sum_{i=1}^{l} y_i x_i \alpha_i = 0$$

es decir,

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

Sustituyendo el valor ínfimo con respecto de $\mathbf{w}$ sobre la definición de la función Lagrangiana, definición 3.6, obtenemos que:

$$\inf_{\mathbf{w}} L(\mathbf{w}, b, \alpha) = -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j < x_i, x_j > \alpha_i \alpha_j + \sum_{j=1}^{l} \alpha_j - b(\sum_{i=1}^{l} y_i \alpha_i)$$

Por tanto, aplicando ahora el mínimo sobre ambas variables $\mathbf{w}, b$, obtenemos:

$$\inf_{\mathbf{w},b} L(\mathbf{w}, b, \alpha) = \begin{cases} -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j < x_i, x_j > \alpha_i \alpha_j + \sum_{j=1}^{l} \alpha_j, si \sum_{i=1}^{l} y_i \alpha_i = 0 \\ -\infty, \text{ otro caso} \end{cases}$$

Juntando todo, llegamos a la expresión del problema que aparece en el enunciado del teorema.

□

El problema dual (3.7) es un problema de maximización, el cual puede ser sustituido por su problema de minimización equivalente, el cual es el siguiente:

$$\begin{cases} & \min_{\alpha} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j < x_i, x_j > \alpha_i \alpha_j - \sum_{j=1}^{l} \alpha_j \\ t.q. & \\ & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & \alpha_i \geq 0, i = 1, ..., l \end{cases} \tag{3.8}$$

Dicho problema de minimización tiene la misma solución que el problema de maximización (3.7) .

Teorema 3.4. *El problema de optimización* (3.8) *es una programación cuadrática convexa.*

Demostración. Fijemos

$$H = (y_i y_j < x_i, x_j >)_{lxl}, \mathbf{e} = (1, ..., 1)^T, \alpha = (\alpha_1, ..., \alpha_l)^T, \mathbf{y} = (y_1, ..., y_l)^T \tag{3.9}$$

entonces, el problema (3.8) puede reescribirse como:

$$\begin{cases} & \text{mín}_\alpha \, W(\alpha) = \frac{1}{2}\alpha^T H\alpha - \mathbf{e}^T\alpha \\ t.q. & \\ & \alpha^T\mathbf{y} = 0 \\ & \alpha \geq 0 \end{cases} \tag{3.10}$$

Fijando $Q = (y_1x_1, ..., y_lx_l)$. Es claro que $H = Q^TQ$ y por tanto H es semidefinida positiva. Por tanto el problema anterior es un problema de programación cuadrática convexa por la definición 2.15 y el teorema 2.2.

□

Teorema 3.5. *Considerando el problema linealmente separable. Para cualquier solución del problema dual* (3.8) *,* $\alpha^* = (\alpha_1^*, ..., \alpha_l^*)^T$*, debe tener una componente* α_j^* *no nula. Además, para cada componente no nula* α_j^* *de* α^**, la única solución del problema de optimización primario* (3.1) *puede ser obtenida de la siguiente forma:*

$$w^* = \sum_{i=1}^{l} \alpha_i^* y_i x_i \tag{3.11}$$

$$b^* = y_j - \sum_{i=1}^{l} \alpha_i^* y_i < x_i, x_j > \tag{3.12}$$

Demostración. En primer lugar, probemos que para el w^* definido por (3.11), existe un $\tilde{b^*}$, de forma que $(w^*, \tilde{b^*})$ sea solución del problema primario. el teorema 3.4 muestra que el problema (3.8) puede ser escrito en la forma (3.10). De esta forma es sencillo probar que este último problema cumple la condición de Slater, definición 2.17. Por tanto, si α^* es una solución del problema (3.10) por tanto se deduce que existe un multiplicador $\tilde{b^*}$ y un vector multiplicador s^* de forma que:

$$\alpha^{*T} y = 0, \alpha^* \geq 0 \tag{3.13}$$

$$s^{*T}\alpha^* = 0, s^* \geq 0 \tag{3.14}$$

$$H\alpha^* - e + \tilde{b^*}y - s^* = 0 \tag{3.15}$$

De donde se obtiene que

$$H\alpha^* - e + \tilde{b^*}y \geq 0 \tag{3.16}$$

De (3.11), se tiene que es equivalente a:

$$y_i(< w^*, x_i > + b^*) \geq 1, i = 1, ..., l \tag{3.17}$$

lo cual implica que $(w^*, \tilde{b^*})$ sea una solución factible del problema primario. Además, de (3.13) - (3.15), obtenemos que:

$$\begin{aligned}\frac{1}{2}\|w^*\|^2 &= \frac{1}{2}\alpha^{*T}H\alpha^* \\ &= \frac{1}{2}\alpha^{*T}H\alpha^* - \alpha^{*T}(H\alpha^* - e + \tilde{b^*}y - s^*) \\ &= -\frac{1}{2}\alpha^{*T}H\alpha^* - \tilde{b^*}\alpha^{*T}y + e^T\alpha^* + s^{*T}\alpha^* \\ &= -\frac{1}{2}\alpha^{*T}H\alpha^* + e^T\alpha^*\end{aligned} \tag{3.18}$$

Con esto queda probado que el valor de la función objetivo del problema primario en el punto $(w^*, \tilde{b^*})$ es igual al valor óptimo del problema dual y por tanto es solución del problema primario.

Ahora veamos que α^* no es nulo. Si fuera nulo, se tendría que w^*, definido en (3.11) sería el vector nulo, lo cual contradice la conclusión de (i) del teorema 3.1 y por tanto, concluimos que $\alpha^* \neq 0$

Por último, veamos que (w^*, b^*) con la forma de (3.11) y (3.12) es la única solución del problema primario. De hecho, la unicidad se deduce del teorema 3.2, por tanto basta con probar que $\tilde{b^*}$ es de la forma de (3.12). Dado que si $\alpha_j^* \neq 0$ implica que $s_j^* = 0$ de (3.14). Por tanto, tenemos en (3.15) la j-essima entrada de $H\alpha^* - e + \tilde{b^*}y$ es 0, resolviendo la ecuación llegamos a que coincide la expresión de $\tilde{b^*}$

□

El teorema anterior nos proporciona una forma de construir la función de decisión, haciendo uso del problema de optimización dual. Para ello, a partir de una solución $\alpha^* = (\alpha_1^*, ..., \alpha_l^*)^T$ del problema (3.8), podemos encontrar una solución (w^*, b^*) del problema primario, haciendo uso de (3.11) y (3.12). Por tanto juntando todo podemos definir el siguiente algoritmo de clasificación basándonos en el problema de optimización dual.

Algoritmo 3.2. *(Algoritmo de clasificación de máquina de vectores soporte dual).*

(1)Introducir el conjunto de entrenamiento $\mathbf{X} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_n}, y_n), ..., (\mathbf{x_N}, y_N)\} \in (\mathbb{R}^D \times \Theta)^N$*,donde* $\mathbf{x_i} \in \mathbb{R}^D, y_i \in \Theta = \{1, -1\}, i = 1, ..., D.$

(2) Construir y resolver el problema de programación cuadrática convexa (3.8)*. Véase la sección 2.2.1.*

$$\begin{cases} \min_\alpha \frac{1}{2}\sum_{i=1}^l \sum_{j=1}^l y_i y_j < x_i, x_j > \alpha_i\alpha_j - \sum_{j=1}^l \alpha_j \\ t.q. \\ \sum_{i=1}^l y_i\alpha_i = 0 \\ \alpha_i \geq 0, i = 1, ..., l \end{cases}$$

obteniendo la solución $\alpha^* = (\alpha_1^*, ..., \alpha_N^*)^T$

(3)Calcular $w^* = \sum_{i=1}^l \alpha_i^* y_i x_i$ *Escoger una componente no nula de* α^*, α_j^* *y calcular* b^*

$$b^* = y_j - \sum_{i=1}^{l} \alpha_i^* y_i < x_i, x_j >$$

(4) Construir el hiperplano separador $< w^*, x > + b^* = 0$ *y su función de decisión asociada*

$$f(x) = sgn(g(x))$$

donde

$$g(x) =< w^*, x > + b^* = \sum_{i=1}^{l} y_i \alpha_i^* < x_i, x > + b^* \tag{3.19}$$

3.4. Vectores soporte

Una vez estudiado cual es el algoritmo a seguir para obtener el hiperplano separador, así como el método de clasificación, vamos a dar sentido al nombre que recibe el algoritmo en sí, es decir, vamos a definir qué son los vectores soporte.

Para ello, podemos observar cómo en los pasos (3) y (4) del algoritmo anterior para determinar el hiperplano separador, y en consecuencia la función de decisión o predictora, se hace uso de un subconjunto de elementos del conjunto de datos de entrenamiento que disponemos, en concreto, el subconjunto está formado por los puntos de entrenamiento correspondientes a las componentes no nulas (positivas) de α. El resto de puntos del conjunto de entrenamiento como hemos visto no juegan ningún papel a la hora de determinar cual es el hiperplano separador y la función de decisión. Con todo esto, damos la siguiente definición.

Definición 3.3. *(Vectores soporte) Supongamos que* $\alpha^* = (\alpha_1^*, ..., \alpha_N^*)^T$ *es solución del problema dual que se obtiene al aplicar el algoritmo 3.2 en el paso (2), el dato de entrada* $\mathbf{x_i}$ *asociado al punto del conjunto de entrenamiento* $(\mathbf{x_i}, y_i)$*, se dice que es un vector soporte si la correspondiente componente* α_i^* *de* α^* *es no nula; en caso de que sea nula no es un vector soporte.*

Con esta definición, podemos decir que en el algoritmo anterior, algoritmo 3.2, la función de decisión y el hiperplano separador es determinado solo a partir de los vectores soporte, lo cual aclara el nombre que recibe el algoritmo como máquina de vectores soporte. Veamos a continuación un teorema que caracteriza a los vectores soporte.

Teorema 3.6. *Supongamos que los problemas linealmente separables han sido resueltos usando el algoritmo 3.2 y que* $g(\mathbf{x})$ *es definida como en* (3.19)*,entonces:*

(i) Los vectores soporte $\mathbf{x_i}$ *satisfacen*

$$y_i g(\mathbf{x_i}) = y_i(< \mathbf{w}^*, \mathbf{x_i} > + b^*) = 1$$

es decir, todos los vectores soporte se encuentran en los dos hiperplanos soporte.

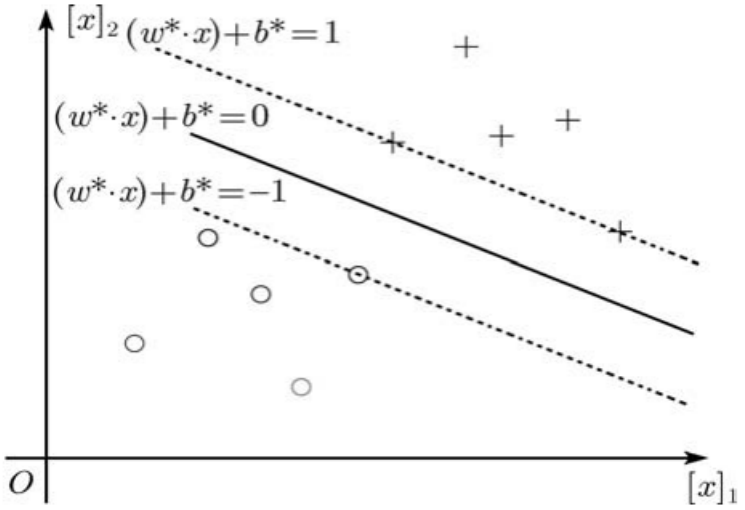


Figura 3.6: Vectores soporte

(ii) Los vectores $\mathbf{x_i}$ *que no son vectores soporte satisfacen*

$$y_i g(\mathbf{x_i}) = y_i(< \mathbf{w}^*, \mathbf{x_i} > +b^*) \geq 1$$

Demostración. Para la demostración debemos de volver a hacer uso de (3.13) - (3.15). Del hecho de que $\alpha^* \geq 0$, tenemos que $s_i^*\alpha_i^* = 0, i = 1, ..., N$. Luego de (3.15) obtenemos que

$$s_i^* = y_i(< w_i^*, x_i > +b^*) - 1, i = 1, ..., N \tag{3.20}$$

Donde, por tanto,

$$s_i^*\alpha_i^* = \alpha_i^*(y_i(< w_i^*, x_i > +b^*) - 1) = 0, i = 1, ..., N \tag{3.21}$$

Por tanto (i) se deduce de (3.21) y del hecho de que $\alpha_i^* \neq 0$ asociado con el vector soporte $\mathbf{x_i}$. (ii) se deduce de (3.20) y del hecho de que en (3.14) $s^* \geq 0$.

□

Para entender cual es la interpretación del anterior teorema ,vamos a hacer uso de la figura 3.6 , en la cual $< \mathbf{w}^*, \mathbf{x} > +b^* = 0$ se corresponde con el hiperplano separador buscado por medio del algoritmo ya comentado, por otra parte, las rectas $< \mathbf{w}^*, \mathbf{x} > +b^* = 1$ y $< \mathbf{w}^*, \mathbf{x} > +b^* = -1$ son los dos hiperplanos soporte. Como podemos observar en la imagen, sobre los hiperplanos soporte se encuentran los vectores soporte de ambas clases, la positiva (con símbolo +) y la negativa con (símbolo ∘).

Por otra parte, aquellos vectores que no son vectores soporte se encuentran a un lado u otro de cada uno de los hiperplanos soporte, para los vectores de la clase positiva se encuentran "por encima"del hiperplano soporte $< \mathbf{w}^*, \mathbf{x} > +b^* = 1$. Los vectores de la clase no negativa "por debajo"del hiperplano soporte $< \mathbf{w}^*, \mathbf{x} > +b^* = -1$.

Capítulo 4

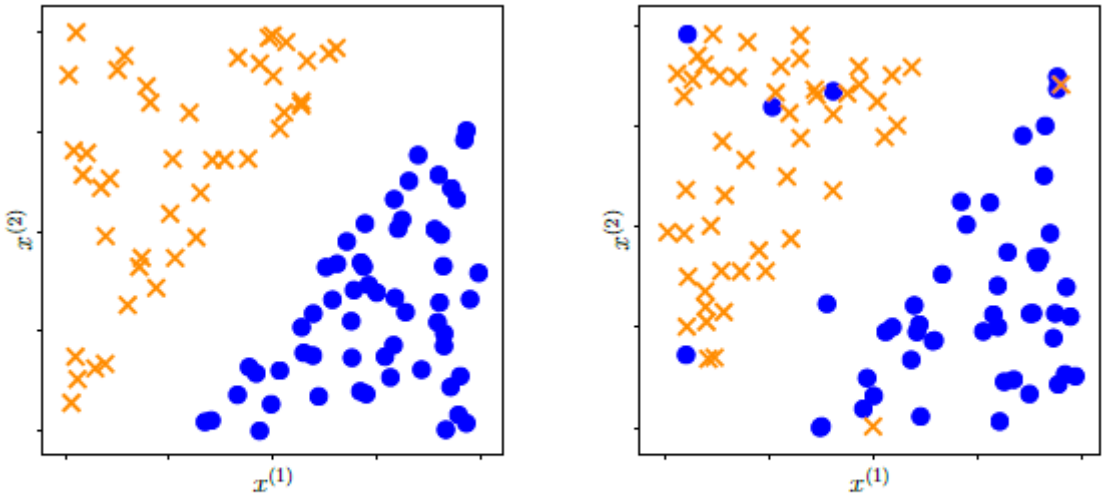
Clasificación lineal con M.V.S. para problemas no linealmente separables. M.V.S. de margen suave

En este apartado de la memoria, vamos a estudiar el caso en el que el conjunto de datos de entrenamiento no son linealmente separables, es decir, no existe un separador lineal que clasifique el conjunto de entrenamiento en dos grupos bien definidos de forma total para todos los puntos. Aún así vamos a tratar de buscar un separador lineal que los clasifique de la mejor forma posible, asumiendo que puede haber errores en la clasificación del conjunto de entrenamiento debido a su carácter de no linealmente separables.

Por tanto, debemos de tener presente que vamos a tener que permitir que algunos de los elementos del conjunto de entrenamiento puedan encontrarse en la región del margen o incluso encontrarse en el lado que delimita el hiperplano separador erróneo (es decir en el lado que agrupa a la mayoría de los datos de la clase contraria a la suya). Para poder visualizarlo de una forma más gráfica se incluye la figura 4.1:

El modelo que vamos a estudiar en este apartado, y que permite algunos errores de clasificación sobre el conjunto de datos es el conocido como Máquina de Vectores Soporte (M.V.S.) de Margen Suave.

Como hemos estudiado en el caso linealmente separable, vamos a hacer un estudio del modelo desde dos puntos de vista: el primero de ellos se corresponde con la M.V.S. de margen suave primaria, derivando de forma geométrica la forma de obtener el margen máximo. Derivaremos la M.V.S. de margen suave primaria desde otro punto de vista por medio de la idea de función de pérdida. Por último, como en el caso linealmente separable, usaremos multiplicadores de Lagrange y derivaremos la M.V.S. de margen suave dual.



(a) Linearly separable data, with a large margin

(b) Non-linearly separable data

Figura 4.1: Datos linealmente separables y no linealmente separables

4.1. M.V.S. de Margen Suave Primaria. Enfoque geométrico.

Como hemos comentado en este caso partimos de un conjunto de entrenamiento que no es linealmente separable, por lo tanto, ningún hiperplano es capaz de separar todos los ejemplos del conjunto de datos en cada una de las dos clases, positiva y negativa. Si queremos de todas formas utilizar un hiperplano como separador, asumiendo los posibles errores que van a existir con alguno de los ejemplos, vamos a tener que tener en cuenta dos estrategias con la intención de que dicho hiperplano pueda separar de la mejor forma a pesar de la no separabilidad de los datos.

La primera de las estrategias a utilizar es considerar variables de holgura, con el fin de relajar las restricciones de que el hiperplano separe de forma correcta todos los ejemplos del conjunto de entrenamiento. De esta forma van a existir ejemplos $\mathbf{x_i}$ en el conjunto de datos que no van a cumplir las restricciones $y_i(< \mathbf{w}, \mathbf{x_i} > +b) \geq 1$. En efecto, se introducen dichas variables de holgura ξ_i

$$\xi_i \geq 0, i = 1, ..., N \tag{4.1}$$

y suavizamos así las restricciones, permitiendo que sean más laxas de la siguiente forma:

$$y_i(< \mathbf{w}, \mathbf{x_i} > +b) \geq 1 - \xi_i, i = 1, ..., N \tag{4.2}$$

De esta manera, restamos el valor de ξ_i del margen, restringiendo que ξ_i sea no negativo. Las variables de holgura ξ_i miden la distancia de cada uno de los ejemplos a su hiperplano soporte correspondiente. En el caso de la figura 4.2, ξ mide la distancia del ejemplo positivo

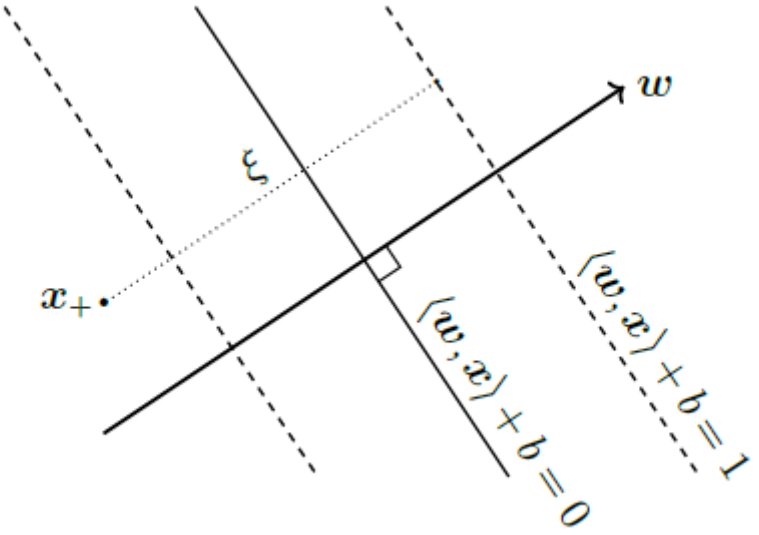


Figura 4.2: Variables de holgura

del conjunto de datos $\mathbf{x}_+$ al hiperplano soporte positivo $y_i(< \mathbf{w}, \mathbf{x} > +b) = 1$, cuando $\mathbf{x}_+$ se encuentra en el lado erróneo de la clasificación.

La otra estrategia que debemos de considerar es intentar de hacer que los respectivos ξ_i sean lo mas pequeños posibles, con la finalidad de que para aquellos ejemplos en los que se no se cumplen las restricciones, dichas variables de holgura sean tan pequeñas como se pueda para que no haya ejemplos del conjunto de entrenamiento en los que haya un gran error en la clasificación de estos (no haya una distancia muy grande de ese ejemplo a la clase a la cual debe permanecer). Para ello, impondremos una penalización sobre la función objetivo del problema de optimización. Por ejemplo, una de las forma de actuar y añadir dicha penalización a la función objetivo, consiste en sumar el término $\sum_{i=1}^{N} \xi_i$ a dicha función objetivo, de esta forma, obtenemos un nuevo problema de optimización, cambiando el problema de optimización primario al siguiente:

$$\begin{cases} & \text{mín}_{\mathbf{w},b,\xi}\ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ t.q. & \\ & y_i(< \mathbf{w}, \mathbf{x_i} > +b) \geq 1 - \xi_i, i = 1, ..., N \\ & \xi_i \geq 0, i = 1, ..., N \end{cases} \tag{4.3}$$

donde, tenemos que $\xi = (\xi_1, ..., \xi_N)^T$, y $C > 0$ es un parámetro de penalización.

El parámetro C pondera el tamaño del margen y la suma de la holgura total que tenemos. Este parámetro, es un parámetro de regularización, el término $\sum_{i=1}^{N} \xi_i$ es el término de regularización tal y como se ha comentado en los conceptos previos a cerca de los conceptos relacionados con el Machine Learning. Este término permite que el predictor no sufra de sobreajuste, que es básicamente lo que estamos tratando en esta sección, que nuestro predictor no tiene porque clasificar de forma correcta todos los datos del conjunto de entrenamiento. En este caso, un valor alto del parámetro C implica una regularización baja, es decir, le estamos dando a las variables de holgura un peso elevado, y estamos dando prioridad a los

ejemplos que no se encuentran en el lado correcto del margen.

Veamos ahora el significado que tiene la función objetivo en el problema de optimización. En (4.3) minimizar $\|\mathbf{w}\|^2$ supone que se maximice el margen, que es el objetivo buscado desde el principio, pero en el caso en el que nos encontramos ahora no solo buscamos que se minimice $\|\mathbf{w}\|^2$ (maximizar margen), sino que también queremos minimizar $\sum_{i=1}^{N} \xi_i$, lo cual es una medida de las restricciones que no se cumplen $y_i(< \mathbf{w}, \mathbf{x_i} > +b) \geq 1, i = 1, ..., N$, tratando que sean lo menor posibles. Por otra parte en (4.3) también podemos ver C como un parámetro de peso entre ambos miembros de la función objetivo, donde dependiendo de su valor se dará mayor peso a maximizar el margen o a minimizar la medida de las restricciones que no se cumplen.

Con todo esto, obtenemos un algoritmo de clasificación por medio de las máquinas de vectores soporte de margen suave, que es una mejora del algoritmo 3.1 de maximización del margen para problemas linealmente separables.

Algoritmo 4.1. *(Método de maximización del margen).*

(1) Introducir el conjunto de entrenamiento

$$\mathbf{X} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_n}, y_n), ..., (\mathbf{x_N}, y_N)\} \in (\mathbb{R}^D \times \Theta)^N$$

donde $\mathbf{x_i} \in \mathbb{R}^D, y_i \in \Theta = \{1, -1\}, i = 1, ..., D$.

(2)Construir y resolver el problema de optimización (4.3) *obteniendo la solución* $(\mathbf{w}^*, b^*, \xi^*)$

(3) Construir el hiperplano separador $< \mathbf{w}^*, \mathbf{x} > +b^* = 0$ *y la función de decisión o estimador* $f(\mathbf{x}) = sgn(< \mathbf{w}^*, \mathbf{x} > +b^*)$

4.2. Propiedades del método de margen máximo

Veamos que la función de decisión del algoritmo de clasificación (así como el hiperplano separador) tan solo dependen de $(\mathbf{w}^*, b^*)$ de la solución total del problema primario (4.3) $(\mathbf{w}^*, b^*, \xi^*)$. Por tanto vamos a considerar el problema primario (4.3) y su solución con respecto a las variables $(\mathbf{w}, b)$.

Teorema 4.1. *Existen soluciones del problema primario* (4.3) *con respecto a las variables* $(\mathbf{w}, b)$.

Demostración. La demostración es similar a la del teorema 3.1, para ello, tomemos $\tilde{\mathbf{w}}, \tilde{b}$ arbitrarios y construyamos $\tilde{\xi} = (\tilde{\xi_1}, ..., \tilde{\xi_N})^T$ de la siguiente forma

$$\tilde{\xi_i} = \text{máx}\{1 - y_i(< \tilde{\mathbf{w}}, \mathbf{x_i} > +\tilde{b}), 0\} \tag{4.4}$$

de manera, que podemos ver que $(\tilde{\mathbf{w}}, \tilde{b}, \tilde{\xi})$ es un punto factible del problema primario. Es más, podemos construir un problema de optimización equivalente al problema primario (4.3):

$$\begin{cases} & \text{mín}_{\mathbf{w},b,\xi}\ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ t.q. & \\ & y_i(<\mathbf{w},\mathbf{x_i}>+b) \geq 1-\xi_i, i=1,...,N \\ & \xi_i \geq 0, i=1,...,N \\ & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \leq \frac{1}{2}\|\tilde{\mathbf{w}}\|^2 + C\sum_{i=1}^{N}\tilde{\xi}_i \end{cases} \tag{4.5}$$

Donde ahora es sencillo probar la existencia de la solución al buscar el mínimo de una función continua sobre un conjunto convexo, cerrado y acotado.

□

Teorema 4.2. *La solución* $\mathbf{w}^*$ *del problema primario* (4.3) *con respecto a la variable* ($\mathbf{w}$ *es única.*

Demostración. La demostración es inmediata a partir del Teorema 1.2.15. de [7]. □

Teorema 4.3. *El conjunto de soluciones del problema primario* (4.3) *con respecto a la variable* b *es un intervalo cerrado y acotado* $[\underline{b}, \overline{b}]$*, donde* $\underline{b} \leq \overline{b}$*.*

Demostración. Por el teorema 4.1, sabemos que el conjunto de soluciones del problema (4.3) con respecto de la variable b es no vacío, puesto que conocemos que el conjunto de soluciones del problema de optimización es un conjunto cerrado y acotado, al igual que la región factible del problema. Esto hace que el conjunto de soluciones debe estar acotado en un intervalo cerrado, ya que las soluciones con respecto a b están acotadas y estamos en $\mathbb{R}$. □

4.3. M.V.S. Suave Dual. Enfoque a través de la función de pérdida

En esta sección vamos a dar el otro enfoque a través del cual vamos a derivar la MVS de margen suave, para ello, haremos uso de la función de pérdida, de la cual ya hemos hablado en qué consiste el método, en el apartado de cuestiones previas a cerca del Machine Learning. Para ello, usaremos el principio de reducción del riesgo empírico.

Para el problema concreto de M.V.S., disponemos de hiperplanos como la clase de hipótesis sobre la que se encuentran parametrizados todos los hiperplanos posibles a escoger en el problema de clasificación. Esto es, que la clase de hipótesis es la siguiente:

$$f(\mathbf{x}) = <\mathbf{w},\mathbf{x}> + b$$

Ya hemos visto que el margen se corresponde con el término de regularización el cual queremos maximizar. Por tanto, el siguiente paso es definir una función de pérdida. Dado que estamos en el caso de la clasificación binaria, la salida del predictor $f(\mathbf{x})$ se corresponde con una de las dos etiquetas $y_i = \{1, -1\}$, por lo que tenemos que considerar una función de pérdida que se adecue al problema de clasificación binaria.

En este caso, la función de pérdida ideal sería aquella que cuente el número de errores existentes entre las etiquetas y las predicciones hechas por la función de predicción sobre los

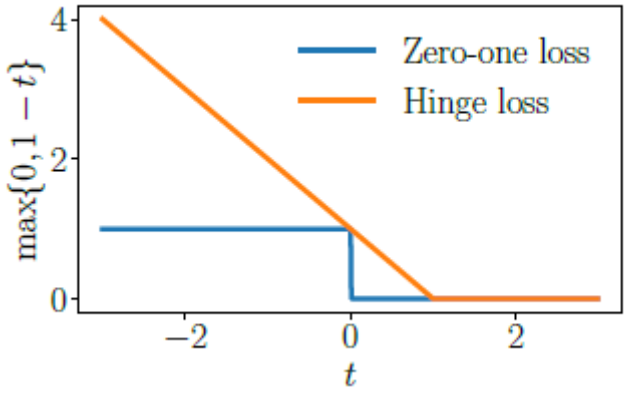


Figura 4.3: Funciones de pérdida

ejemplos del conjunto de entrenamiento. Es decir, para el predictor f aplicado al ejemplo del conjunto de entrenamiento $\mathbf{x_i}$, comparamos la salida del predictor $f(\mathbf{x_i})$ con la etiqueta y_i, y se define la función de pérdida como cero si ambos coinciden y uno si no coinciden. A esta función de perdida se la denota como $\mathbf{1}(f(\mathbf{x_i}) \neq y_i)$ y se llama función de pérdida 1-0. El problema que presenta esta función de pérdida es que el resultado es un problema de optimización combinatorio, el cual es mucho más complicado de resolver en comparación con los problemas de optimización continuos.

Por tanto, la estrategia a tomar en este caso consiste en escoger una función de pérdida continua, para que el problema sea más sencillo. Para ello, utilizaremos la función de pérdida hinge, cuya definición es la siguiente:

$$l(t) = \text{máx}\{0, 1 - t\}, \text{donde } t = yf(\mathbf{x}) = y(< \mathbf{w}, \mathbf{x} > + b) \tag{4.6}$$

A partir de esto, si $f(\mathbf{x})$ se encuentra en el lado correcto (basándonos en su correspondiente etiqueta y) del hiperplano separador, y se encuentra a una distancia mayor que 1, esto implica que $t \geq 1$ y por tanto la función de pérdida hinge tiene valor de 0. Por otra parte, si $f(\mathbf{x})$ se encuentra en el lado correcto, pero demasiado cerca del hiperplano separador ($0 < t < 1$), el ejemplo $\mathbf{x}$ se encuentra en el margen, y en consecuencia la función de pérdida devuelve un valor positivo y menor que 1. Para el caso en el que el ejemplo se encuentra en el lado incorrecto del hiperplano, es decir, $t < 0$, la función de pérdida hinge devolverá un número positivo y grande (mayor que 1), que se incrementará de forma lineal.

Una alternativa es mostrar la función de pérdida como dos porciones lineales de la siguiente forma:

$$l(t) = \begin{cases} 0 \text{ , si } t \geq 1 \\ 1 - t \text{ ,si } t < 1 \end{cases} \tag{4.7}$$

Podemos visualizar la función de pérdida hinge y la función de pérdida 1-0 en la figura 4.3.

Con todo esto, nuestro propósito es minimizar la pérdida total para un cierto conjunto de entrenamiento dado. Para ello, haciendo uso de la función de perdida hinge, nos encontramos en el siguiente problema de optimización sin restricciones.

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max\{0, 1 - y_i(< \mathbf{w}, \mathbf{x_i} + b)\} \tag{4.8}$$

El primero de los términos, como ya hemos comentado anteriormente, es el término de regularización, ,y trata de maximizar el margen. El segundo de los términos se corresponde con el término de pérdida o término de error.

A continuación, vamos a ver como el problema de optimización sin restricciones (4.8) es equivalente al problema de optimización con restricciones del enfoque geométrico (4.3). Para ello, nos fijamos en la función de pérdida hinge (4.6), la cual consta de dos partes lineales como se ha expresado en (4.7). Considerando la función de pérdida hinge como un par ejemplo-etiqueta, podemos suprimir la minimización de la función de pérdida hinge a través de t de forma equivalente por la minimización de las variables de holgura ξ con dos restricciones. Es decir, podemos sustituir de forma equivalente

$$\min_{t} \max\{0, 1 - t\}$$

por:

$$\begin{cases} & \min_{\xi,t} \xi \\ t.q. & \\ & \xi \geq 0, \xi \geq 1 - t \end{cases}$$

Sustituyendo esta última expresión en (4.8) y ordenando las restricciones tenemos la misma expresión exactamente que cuando hemos obtenido el problema de optimización para el margen suave desde el punto de vista geométrico (4.3).

4.4. M.V.S. de Margen Suave Dual

Al igual que en el caso linealmente separable del problema de optimización primario (4.3), podemos derivar su problema de optimización dual. Los pasos a seguir aquí serán similares a los tomados en el caso linealmente separable, así como la justificación de por qué utilizar un método u otro, por lo que pasaremos con mayor ligereza sobre este apartado.

En primer lugar, definiremos el problema dual; para ello, la función lagrangiana correspondiente al problema primario (4.3) es :

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i(y_i(< \mathbf{w}, \mathbf{x_i} > +b) - 1 + \xi_i) - \sum_{i=1}^{N} \beta_i\xi_i \tag{4.9}$$

donde $\alpha = (\alpha_1, ..., \alpha_N)^T$ y $\beta = (\beta_1, ..., \beta_N)^T$ son los vectores de los multiplicadores de Lagrange. Con esto, tenemos el siguiente teorema:

Teorema 4.4. *El problema de optimización:*

$$\begin{cases} & \text{máx}_{\alpha,\beta} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j < \mathbf{x_i}, \mathbf{x_j} > + \sum_{j=1}^{N} \alpha_j \\ t.q. & \\ & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & C - \alpha_i - \beta_i = 0, i = 1, ..., N \\ & \alpha_i \geq 0, i = 1, ..., N \\ & \beta_i \geq 0, i = 1, ..., N \end{cases} \tag{4.10}$$

es el problema dual del problema primario (4.3).

Demostración. Derivemos la función lagrangiana (4.9) con respecto a las tres variables primarias, es decir $\mathbf{w}, b, \xi$, obteniendo:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w}^T - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}^T \tag{4.11}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i \tag{4.12}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i \tag{4.13}$$

Para encontrar el máximo de la función lagrangiana, debemos de igualar cada una de las derivadas parciales a 0. De esta forma, igualando (4.11) a cero, tenemos que:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}$$

Esto indica, como ya habíamos comentado en el caso lineal, que el vector de pesos óptimo es una combinación lineal de ejemplos del conjunto de entrenamiento $\mathbf{x_i}$. Más en concreto, igualando (4.12) a 0 implica que realmente el vector de pesos óptimo es una combinación afín de ejemplos del conjunto de entrenamiento.

Llegados a este punto, sustituyendo la expresión de $\mathbf{w}$ sobre la función lagrangiana (4.9), obtenemos el dual.

$$\begin{aligned} &\tfrac{1}{2}\textstyle\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j < \mathbf{x_i}, \mathbf{x_j} > + \sum_{i=1}^{N} y_i \alpha_i < \sum_{j=1}^{N} y_j \alpha_j \mathbf{x_j}, \mathbf{x_i} > + \\ &C\textstyle\sum_{i=1}^{N} \xi_i - b\sum_{i=1} y_i \alpha_i + \sum_{i=1} \alpha_i - \sum_{i=1} \alpha_i \xi_i - \sum_{i=1} \beta_i \xi_i \end{aligned} \tag{4.14}$$

De esta forma, no existen términos que involucren la variable primaria $\mathbf{w}$. De igualar (4.12) a 0, obtenemos que $\sum_{i=1}^{N} y_i \alpha_i = 0$. Por tanto, el término que involucra a b también desaparece.

Del hecho de que el producto interno sea bilineal y simétrico, tenemos que los dos primeros términos de (4.14) son sobre los mismos elementos, los cuales pueden ser simplificados, obteniendo el lagrangiano:

$$L(\xi, \alpha, \beta) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j < \mathbf{x_i}, \mathbf{x_j} > + \sum_{i=1}^{N} \alpha_i + \sum_{i=1}^{N}(C - \alpha_i - \beta_i)\xi_i \tag{4.15}$$

El último de los términos reúne las variables de holgura. Igualando (4.13) a cero, vemos que el último de los términos de (4.15) es cero. Usando la misma ecuación y recordando que β_i son no negativos, tenemos que $\alpha_i \leq C$.

Reorganizando todo lo que tenemos hasta ahora, queremos maximizar la función lagrangiana (4.15) con respecto a las condiciones que hemos ido imponiendo al igualar las derivadas parciales a 0, con lo cual llegamos al problema de optimización del enunciado. □

Cabe destacar que el problema dual (4.10) puede ser simplificado a un problema solo en las variables α, eliminando la variable β y reescribiendo el problema como un problema de minimización. Usamos la observación hecha en la demostración del teorema anterior de que $\alpha_i \leq C$:

$$\begin{cases} & \text{mín}_\alpha \, \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j < \mathbf{x_i}, \mathbf{x_j} > - \sum_{j=1}^{N} \alpha_j \\ t.q. & \\ & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, ..., N \end{cases} \tag{4.16}$$

Enunciemos un último teorema:

Teorema 4.5. *Supongamos que $\alpha^* = (\alpha_1^*, ..., \alpha_N^*)$ es una solución del problema de programación cuadrática convexa* (4.16)*. Si existe una componente de α^*, α_j^* de forma que $\alpha_j^* \in (0, C)$, entonces una solución $(\mathbf{w}^*, b^*)$ del problema primario* (4.3) *con respecto a las variables $(\mathbf{w}, b)$ puede ser obtenida por:*

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x_i}$$

$$b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i < \mathbf{x_i}, \mathbf{x_j} >$$

Demostración. Es similar a la realizada en el teorema 3.5 □

Con todos estos teoremas podemos establecer el siguiente algoritmo de clasificación por medio de MVS de margen suave.

Algoritmo 4.2. *(Clasificación lineal de margen suave por vectores soporte).*

(1) Introducir el conjunto de entrenamiento

$$\mathbf{X} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_n}, y_n), ..., (\mathbf{x_N}, y_N)\} \in (\mathbb{R}^D \times \Theta)^N$$

donde $\mathbf{x_i} \in \mathbb{R}^D, y_i \in \Theta = \{1, -1\}, i = 1, ..., D$.

(2) Elegir un parámetro de penalización apropiado $C > 0$.

(3)Construir y resolver el problema de optimización convexo y cuadrático (4.16). *Véase la sección 2.2.1*

$$\begin{cases} & \text{mín}_\alpha \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j < \mathbf{x_i}, \mathbf{x_j} > - \sum_{j=1}^{N} \alpha_j \\ t.q. & \\ & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, ..., N \end{cases}$$

obteniendo la solución $\alpha^* = (\alpha_1^*, ..., \alpha_N^*)^T$

(4) Calcular b^*: Elegir una componente de $\alpha^*, \alpha_j^* \in (0, C)$ y calcular:

$$b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i < \mathbf{x_i}, \mathbf{x_j} >$$

(5) Construir la función de decisión

$$f(\mathbf{x}) = sgn(g(\mathbf{x}))$$

donde

$$g(\mathbf{x}) = \sum_{i=1}^{N} y_i \alpha_i^* < \mathbf{x_i}, \mathbf{x} > + b^*$$

El anterior algoritmo puede ser utilizado tanto para problemas linealmente separables como para problemas que no sean linealmente separables. Luego en el caso linealmente separable, ya hemos visto que el algoritmo 3.2 también sirve. Cuando el parámetro $C \to \infty$ el problema primario (4.3) se reduce al problema primario del caso linealmente separable. Los problemas primarios asociados a ambos algoritmos no son los mismos en general. Por tanto, las funciones de decisión obtenidas no son las mismas. La función de decisión del caso linealmente separable no es necesariamente mejor, pero es diseñada para ese problema específico. El problema de dicho algoritmo se encuentra cuando entre el conjunto de entrenamiento existen puntos que son clasificados de forma errónea, los cuales afectarían al hiperplano separador y por tanto al algoritmo. Sin embargo este último algoritmo que hemos estudiado para el caso no linealmente separable es capaz de solventar este problema hasta cierto punto.

Capítulo 5

Clasificación no lineal en problemas no linealmente separables. Kernels

En esta sección vamos a tratar el caso en el que el conjunto de entrenamiento no es linealmente separable, y no lo es no solo por un reducido número de ejemplos del conjunto de entrenamiento sino que en general no existe un separador lineal que clasifique de forma correcta la mayoría de los ejemplos del conjunto de entrenamiento.

Por tanto, para clasificar los ejemplos del conjunto de entrenamiento en este caso tenemos que recurrir a los separadores no lineales. Para ello va a ser necesario adoptar una serie de técnicas, que se van a desarrollar en la sección. Para entender un poco mejor el caso con el que estamos tratando, vamos a ver un ejemplo en dos dimensiones con la idea de mostrar como se va a generalizar la clasificación lineal en la no lineal.

En el ejemplo siguiente, tratamos un problema de clasificación en el cual los datos del conjunto de entrenamiento no son linealmente separables. Por lo tanto no es posible realizar una clasificación lineal basada en hiperplanos lineales. El conjunto de entrenamiento lo forman 20 puntos en $\mathbb{R}^2$ como se muestra en la figura 5.1. Los puntos + se corresponde con los puntos del conjunto de entrenamiento de la clase positiva, es decir aquellos en los que $y_i = 1$ y los puntos $\circ$ se corresponden con los de la clase negativa $y_i = -1$.

Como podemos ver en la figura 5.1, el separador se corresponde con una linea curva, en concreto una elipse centrada en el origen. Por este motivo, se dice que el separador es no lineal. El problema aquí radica en la forma de encontrar dicha elipse que sirve como separador en el método. Puesto que ya disponemos de un mecanismo de separación lineal, lo más lógico sería intentar adaptar este método para tratar con el caso no lineal en vez de construir un nuevo método para el caso no lineal. Por tanto esta será la forma de actuar.

El punto clave es como transformar la curva (elipse) en una recta. La solución consta en considerar un mapeo, es decir una aplicación (una biyección), que lleve los puntos del plano $[x]_1 0 [x]_2$ en los puntos $(\phi_1(x_1, x_2), \phi_2(x_1, x_2))$ sobre el mismo plano $[x]_1 0 [x]_2$. En el ejemplo concreto que estamos tratando la función $\phi(x_1, x_2) = (\phi_1(x_1, x_2), \phi_2(x_1, x_2))$ se corresponde con:

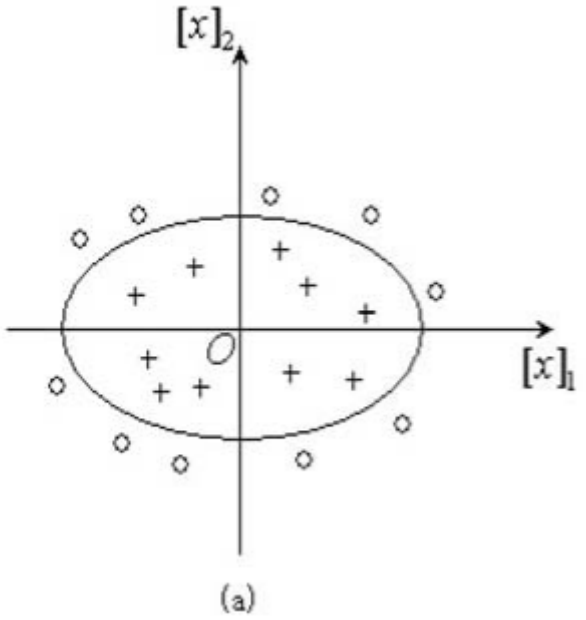


Figura 5.1: Ejemplo no lineal

$$\phi_1(x_1, x_2) = x_1^2, \phi_2(x_1, x_2) = x_2^2 \tag{5.1}$$

Con dicha aplicación lineal podremos mapear o transformar la elipse $\alpha x_1^2 + \beta x_2^2 - r^2 = 0$ en el plano $[x]_1 0 [x]_2$ en la recta $\alpha x_1 + \beta x_2 - r^2 = 0$ sobre el mismo plano. Véase la figura 5.2

Por tanto, una vez aplicado dicho truco, tan solo debemos de aplicar la transformación dada por la aplicación lineal ϕ como en (5.1) sobre las entradas de los datos del conjunto de entrenamiento para obtener sobre el mismo plano un problema que sea linealmente separable. Una vez que disponemos de un problema linealmente separable debemos de aplicar clasificación lineal por medio de M.V.S. como hemos visto en los apartados anteriores para obtener el hiperplano separador sobre el problema lineal transformado. Por último, debemos de calcular la curva que actuará como separador en el problema no transformado a partir de la curva del problema transformado, así como la función de decisión.

Una vez que hemos visto un ejemplo concreto en dos dimensiones, nuestro propósito es llevar a cabo dicho truco de transformar el problema no linealmente separable en uno linealmente separable en cualquier dimensión.

5.1. Máquina de clasificación basada en separación no lineal

Como ya se ha comentado, lo único necesario para poder generalizar el hecho de transforma un problema no linealmente separable en uno linealmente separable es una aplicación que permita mapear los datos. De forma general, dicha aplicación transforma un $\mathbf{x}$ vector D-dimensional en uno M-dimensional en el espacio euclídeo $\mathbb{R}^m$, o incluso en un vector infinito dimensional de un espacio de Hilbert. Dicha aplicación se puede expresar de la siguiente

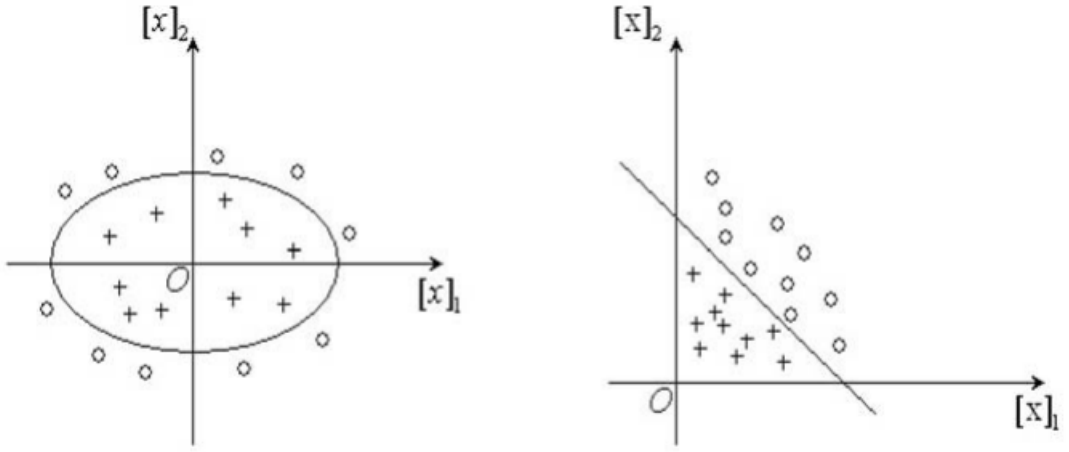


Figura 5.2: Mapeo

forma:

$$\phi : \mathbb{R}^n \to \mathbb{H}, \mathbf{x} = (x_1, ..., x_D)^T \to \mathbf{x} = (x_1, x_2, ...)^T = \phi(\mathbf{x}) \tag{5.2}$$

Si suponemos que el conjunto de entrenamiento original se corresponde con :

$$\mathbf{X} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_N}, y_N)\} \in (\mathbb{R}^D \times \Theta)^N$$

donde $\mathbf{x_i} \in \mathbb{R}^D$, $y_i \in \Theta = \{1, -1\}, i = 1..., N$, después de aplicar la aplicación (5.2) el conjunto de entrenamiento $\mathbf{X}$ se transforma en

$$T_\phi = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_N}, y_N)\}$$

donde $\mathbf{x_i} = \phi(\mathbf{x_i}) \in \mathbb{H}$, $y_i \in \Theta = \{1, -1\}, i = 1..., N$.

El siguiente paso consiste en calcular el hiperplano separador ($< \mathbf{w}^*, \mathbf{x} > +b^* = 0$) en $\mathbb{H}$ y así poder deducir la hipersuperficie de separación $< \mathbf{w}^*, \phi(\mathbf{x}) > +b^* = 0$ y la función de decisión $f(\mathbf{x}) = sgn(< \mathbf{w}^*, \mathbf{x} > +b^*) = sgn(< \mathbf{w}^*, \phi(\mathbf{x}) > +b^*)$ en el espacio original $\mathbb{R}^D$.

Debemos destacar que en el espacio de Hilbert $\mathbb{H}$ la distancia entre ambos hiperplanos soporte $< \mathbf{w}, \mathbf{x} > +b = 1$ y $< \mathbf{w}, \mathbf{x} > +b = -1$ puede seguir siendo representada por $\frac{2}{\|\mathbf{w}\|}$, con lo cual podemos derivar el problema de optimización primario (4.3).

$$\begin{cases} & \text{mín}_{\mathbf{w},b,\xi}\ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^N \xi_i \\ t.q. & \\ & y_i(< \mathbf{w}, \phi(\mathbf{x_i}) > +b) \geq 1 - \xi_i, i = 1, ..., N \\ & \xi_i \geq 0, i = 1, ..., N \end{cases} \tag{5.3}$$

Como se hizo en la sección anterior para el caso de clasificación lineal en problemas no linealmente separables, se deriva el problema de optimización dual siguiente (omitimos los pasos dado que ya han sido realizados en la sección anterior):

Teorema 5.1. *El problema de optimización*

$$\begin{cases} & \text{máx}_{\alpha,\beta} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j < \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) > + \sum_{j=1}^{N} \alpha_j \\ t.q. & \\ & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & C - \alpha_i - \beta_i = 0, i = 1, ..., N \\ & \alpha_i \geq 0, i = 1, ..., N \\ & \beta_i \geq 0, i = 1, ..., N \end{cases} \tag{5.4}$$

es el problema dual del problema primario (5.3)

Siguiendo todos los pasos realizados en la anterior sección, incluyendo los teoremas de existencia de soluciones adaptados a este nuevo problema se obtiene el siguiente algoritmo:

Algoritmo 5.1. *(Clasificación lineal de margen suave por vectores soporte).*

(1) Introducir el conjunto de entrenamiento $\mathbf{X} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)), ..., (\mathbf{x_n}, y_n), ..., (\mathbf{x_N}, y_N)\} \in (\mathbb{R}^D \times \Theta)^N$*,donde* $\mathbf{x_i} \in \mathbb{R}^D, y_i \in \Theta = \{1, -1\}, i = 1, ..., D.$

(2) Elegir una aplicación apropiada $\phi : \mathbf{x} = \phi(\mathbf{x})$ *del espacio* $\mathbb{R}^D$ *al espacio de Hilbert* $\mathbb{H}$ *y un parámetro de penalización* $C > 0$*.*

(3)Construir y resolver el problema de optimización convexo y cuadrático. Véase la sección 2.2.1.

$$\begin{cases} & \text{mín}_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j < \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) > - \sum_{j=1}^{N} \alpha_j \\ t.q. & \\ & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, ..., N \\ & \alpha_i \geq 0, i = 1, ..., N \\ & \beta_i \geq 0, i = 1, ..., N \end{cases}$$

obteniendo la solución $\alpha^* = (\alpha_1^*, ..., \alpha_N^*)^T$

(4) Calcular b^*: Elegir una componente de $\alpha^*, \alpha_j^* \in (0, C)$ y calcular:

$$b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i < \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) >$$

(5) Construir la función de decisión

$$f(\mathbf{x}) = sgn(g(\mathbf{x}))$$

donde

$$g(\mathbf{x}) = \sum_{i=1}^{N} y_i \alpha_i^* < \phi(\mathbf{x_i}), \phi(\mathbf{x}) > + b^*$$

Como se puede comprobar, la única diferencia entre el algoritmo que acabamos de estudiar y el algoritmo 4.2 de clasificación lineal de margen suave para problemas no linealmente separable es que en el que acabamos de estudiar se usa el producto interno $< \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) >$ y $< \phi(\mathbf{x_i}), \phi(\mathbf{x}) >$, y en el algoritmo 4.2 se utiliza el producto interno $< \mathbf{x_i}, \mathbf{x_j} >$ y $< \mathbf{x_i}, \mathbf{x} >$.

5.2. Kernels

Como podemos observar, en el algoritmo anterior, algoritmo 5.1, la aplicación ϕ es utiliza siempre con el producto interno $< \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) >$ y $< \phi(\mathbf{x_i}), \phi(\mathbf{x}) >$, y no aparece de forma independiente. Por tanto, podemos definir una función de la siguiente forma:

$$K(\mathbf{x}, \mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}') > \tag{5.5}$$

De esta forma, en el desarrollo realizado anteriormente, en concreto en el algoritmo, podemos sustituir el producto interno $< \phi(.), \phi(.) >$, por la función $K(.,.)$ y seguiremos teniendo la misma función de decisión. A la función definida en (5.5) se la conoce como función kernel. En esta sección vamos a estudiar las propiedades que tienen las funciones kernel, así como la construcción de los principales kernels.

5.2.1. Propiedades

En primer lugar, vamos a dar una definición formal de kernel.

Definición 5.1. *(Kernel)Una función $K(\mathbf{x}, \mathbf{x}')$ definida en $\mathbb{R}^D \times \mathbb{R}^D$ se llama kernel si existe una aplicación ϕ del espacio $\mathbb{R}^D$ al espacio de Hilbert H*

$$\phi : \mathbb{R}^n \to \mathbb{H}, \mathbf{x} = (x_1, ..., x_D)^T \to \mathbf{x} = (x_1, x_2, ...)^T = \phi(\mathbf{x}) \tag{5.6}$$

de forma que

$$K(\mathbf{x}, \mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}') > \tag{5.7}$$

donde $<,>$ denota el producto interno en el espacio de Hilbert H.

Veamos ahora una definición derivada de obtener en forma matricial la representación de la función kernel.

Definición 5.2. *(Matriz de Gram) Para una función $K(\mathbf{x}, \mathbf{x}') : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ y N puntos $\mathbf{x_1}, ..., \mathbf{x_N} \in \mathbb{R}^D$, la matriz de dimensiones $D \times D$, denotada por K, donde para su elemento en la fila i-essima y columna j-essima, se corresponde con $K_{ij} = K(\mathbf{x_i}, \mathbf{x_j})$, se la llama matriz de Gram de la función $K(\mathbf{x}, \mathbf{x}')$ con respecto a $\mathbf{x_1}, ..., \mathbf{x_N}$.*

Teorema 5.2. *(Propiedades del kernel) Una función simétrica $K(\mathbf{x}, \mathbf{x}')$ definida en $\mathbb{R}^D \times \mathbb{R}^D$ es un kernel, si y solo si, la matriz de Gram de $K(\mathbf{x}, \mathbf{x}')$ con respecto a $\mathbf{x_1}, ..., \mathbf{x_N}$ es semidefinida positiva para todo N y todo $\mathbf{x_1}, ..., \mathbf{x_N}$.*

Demostración. Del teorema se asume que la función $K(\mathbf{x}, \mathbf{x}')$ debe de ser simétrica por la propia construcción, puesto que la función kernel está definida sobre productos internos, los cuales deben de tener la propiedad de simetría.

Si suponemos que $K(\mathbf{x}, \mathbf{x}')$ es un kernel, veamos que es semidefinida positiva. Para ello, supongamos que

$$K_{ij} = K(\mathbf{x_i}, \mathbf{x_j}) =< \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) >$$

con $i, j = 1, ..., N$.

Dado $\alpha \in \mathbb{R}^D$ cualquiera, se tiene:

$$\mathbf{v}'K\mathbf{v} = \sum_{i,j=1}^{D} v_i v_j K_{ij} = \sum_{i,j=1}^{D} v_i v_j < \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) >$$

$$=< \sum_{i=1}^{D} v_i \phi(\mathbf{x_i}), \sum_{j=1}^{D} v_j \phi(\mathbf{x_j}) >$$

$$= \| \sum_{i,j=1}^{D} v_i \phi(\mathbf{x_i}) \|^2 \geq 0$$

La otra implicación requiere mucha teoría y se omite por no extender la memoria, puede verse en [5].

□

5.2.2. Construcción

Una de las cuestiones que han quedado pendientes desde que se ha empezado a desarrollar el método de clasificación no lineal, es la forma de escoger la función ϕ que nos permita llevar nuestro problema no linealmente separable en uno que sea linealmente separable. En todo el desarrollo teórico, hemos supuesto que existe dicha función ϕ que cumple con nuestro propósito. En esta sección vamos a estudiar como obtener la función ϕ, en concreto, vamos a estudiar como construir los kernels, que como ya hemos visto sería equivalente a construir la función ϕ.

Para abordar el problema de definir aquellas funciones que actúan como kernels de todas las posibilidades existentes, vamos a seguir tres pasos:

1. Estudiar los kernels básicos.
2. Estudiar las operaciones que conservan a los kernels.
3. A partir de los kernels básicos, construir nuevos kernels por medio de las operaciones que los mantienen.

En un primer lugar vamos a estudiar aquellos kernels básicos.

Teorema 5.3. *La función $K(\mathbf{x}, \mathbf{x}') =< \mathbf{x}, \mathbf{x}' >$ definida en $\mathbb{R}^D \times \mathbb{R}^D$ es un kernel.*

Demostración. Fijemos $\phi(\mathbf{x}) = \mathbf{x}$, entonces $K(\mathbf{x}, \mathbf{x}')$ puede expresarse como

$$K(\mathbf{x}, \mathbf{x}') =< \mathbf{x}, \mathbf{x}' >=< \phi(\mathbf{x}), \phi(\mathbf{x}') >$$

de acuerdo con la definición de Kernel, queda probado el teorema. □

Teorema 5.4. *Si $f(.)$ es una función real definida en $\mathbb{R}^D$, entonces $K(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$ es un kernel. En particular, la función escalar $K(\mathbf{x}, \mathbf{x}') \equiv a$, donde a es un escalar no negativo es un kernel.*

Demostración. Para cualquier $\mathbf{x_1}, ..., \mathbf{x_N} \in \mathbb{R}^D$, consideramos la matriz de Gram

$$(K(\mathbf{x_i}, \mathbf{x_j}))_{D \times D}$$

de la función $K(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$. Entonces, para cualquier vector $\alpha = (\alpha_1, ..., \alpha_D)^T \in \mathbb{R}^D$, se tiene:

$$\alpha^T (K(\mathbf{x_i}, \mathbf{x_j}))_{DxD} \alpha = \sum_{i=1}^{D} \sum_{j=1}^{D} \alpha_i \alpha_j K(\mathbf{x_i}, \mathbf{x_j}) = \sum_{i=1}^{D} \sum_{j=1}^{D} \alpha_i \alpha_j f(\mathbf{x_i}) f(\mathbf{x_j})$$

$$= \sum_{i=1}^{D} \alpha_i f(\mathbf{x_i}) \sum_{j=1}^{D} \alpha_j f(\mathbf{x_j}) = (\sum_{i=1}^{D} \alpha_i f(\mathbf{x_i}))^2 \geq 0$$

Por tanto, la matriz de Gram es semidefinida positiva, y obviamente es simétrica por la conmutabilidad del producto. Por tanto, por el teorema 5.2, la función $K(\mathbf{x}, \mathbf{x}')$ es un kernel. Particularmente, la conslusión también es cierta para el caso de que $f(\mathbf{x}) \equiv \sqrt{a}$ y probamos que $K(\mathbf{x}, \mathbf{x}') \equiv a$ también es un kernel. □

Una vez que hemos estudiado aquellos kernels más básicos, pasamos a estudiar aquellas operaciones que se pueden realizar sobre ellos, con el fin de que al aplicar dichas operaciones, el resultado sigue siendo un kernel. Es decir, vamos a probar que la clase de kernels es cerrada para las siguientes operaciones.

Teorema 5.5. *Supongamos que $K_1(\mathbf{x}, \mathbf{x}')$ y $K_2(\mathbf{x}, \mathbf{x}')$ son kernels en $\mathbb{R}^D \times \mathbb{R}^D$ entonces, su suma:*

$$K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$$

y su producto

$$K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') K_2(\mathbf{x}, \mathbf{x}')$$

son también kernels.

Demostración. Por el teorema 5.2, tan solo debemos probar que para cualquier conjunto de D puntos $\{\mathbf{x_1}, ..., \mathbf{x_D}\}$ en $\mathbb{R}^D$, las matrices de Gram de $K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$ y de $K_1(\mathbf{x}, \mathbf{x}')K_2(\mathbf{x}, \mathbf{x}')$ con respecto a $\{\mathbf{x_1}, ..., \mathbf{x_D}\}$ en $\mathbb{R}^D$ son semidefinidas positivas.

En primer lugar vamos a hacer la prueba para $K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$, y sean K_1 y K_2 las correspondientes matrices de Gram de $K_1(\mathbf{x}, \mathbf{x}')$ y $K_2(\mathbf{x}, \mathbf{x}')$ respectivamente. Tomemos $\alpha \in \mathbb{R}^D$ cualquiera, entonces

$$\alpha^T (K_1 + K_2) \alpha = \alpha^T K_1 \alpha + \alpha^T K_2 \alpha \geq 0$$

Puesto que $K_1(\mathbf{x}, \mathbf{x}')$ y $K_2(\mathbf{x}, \mathbf{x}')$ son kernels y sus matrices son semidefinidas positivas, luego con esto aseguramos que $K_1 + K_2$ es semidefinida positiva.

Vayamos ahora con el caso de $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}')K_2(\mathbf{x}, \mathbf{x}')$; para ello, supongamos que K es la matriz de Gram de $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}')K_2(\mathbf{x}, \mathbf{x}')$ la cual se la conoce como el

producto de Schur de las matrices de Gram K_1 de $K_1(\mathbf{x}, \mathbf{x}')$ y K_2 de $K_2(\mathbf{x}, \mathbf{x}')$, es decir, el elemento $K_{ij} = K_{1ij}K_{2ij}$. Por tanto probemos que K es semidefinida positiva.

$$K = K_1 \circ K_2 \tag{5.8}$$

Del hecho de que K_1 y K_2 sean semidefinidas positivas y simétricas, podemos obtener la descomposición siguiente, fijando C y D, $K_1 = C^T C, K_2 = D^T D$, entonces para cualquier $\alpha \in \mathbb{R}^D$, se tiene que:

$$\alpha^T(K_1 \circ K_2)\alpha = tr[(diag\alpha)K_1(diag\alpha)K_2^T] = tr[(diag\alpha)C^T C(diag\alpha)D^T D]$$

$$= tr[D(diag\alpha)C^T C(diag\alpha)D^T] = tr[[C(diag\alpha)D^T]^T C(diag\alpha)D^T] \geq 0$$

donde en la tercera igualdad se ha utilizado que $trAB = trBA$ para cualquier matriz A, B, con lo cual queda probado que K es semidefinida positiva.

□

Teorema 5.6. *Supongamos que $K_3(\phi, \phi')$ es un kernel en $\mathbb{R}^m \times \mathbb{R}^m$. Si $\phi(\mathbf{x})$ es una aplicación de $\mathbb{R}^D$ en $\mathbb{R}^m$, entonces $K(\mathbf{x}, \mathbf{x}') = K_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$ es un kernel en $\mathbb{R}^D \times \mathbb{R}^D$. En particular, si B es una matriz $D \times D$ semidefinida positiva, entonces $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T B\mathbf{x}'$ es un kernel en $\mathbb{R}^D \times \mathbb{R}^D$.*

Demostración. Para cualquier conjunto de puntos $\{\mathbf{x_1}, ..., \mathbf{x_D}\}$ en $\mathbb{R}^D$, la correspondiente matriz de Gram de $K(\mathbf{x}, \mathbf{x}') = K_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$ es

$$(K(\mathbf{x_i}, \mathbf{x_j}))_{i,j=1}^D = (K_3(\phi(\mathbf{x_i}), \phi(\mathbf{x_j})))_{i,j=1}^D$$

Denotando por $\phi(\mathbf{x_t}) = \phi_t, t = 1, ..., D$, se tiene que

$$(K(\mathbf{x_i}, \mathbf{x_j}))_{i,j=1}^D = (K_3(\phi_i, \phi_j)))_{i,j=1}^D$$

Puesto que $K_3(\phi, \phi')$ es un kernel, la matriz de la derecha es semidefinida positiva; por tanto la matriz K es semidefinida positiva y se termina la prueba de que $K(\mathbf{x}, \mathbf{x}')$ sea un kernel.

Para el caso particular en el que se considera una matriz semidefinida positiva B, la cual puede ser descompuesta en:

$$B = V^T \Lambda V$$

donde V es una matriz ortogonal y Λ es una matriz diagonal que contiene los autovalores de B. Consideremos $\sqrt{\Lambda}$ la matriz diagonal formada por la raíz cuadrada de los autovalores de B. Definiendo un núcleo $K_3(\phi, \phi') =< \phi, \phi' >$ en $\mathbb{R}^D \times \mathbb{R}^D$ y fijando que $\phi(\mathbf{x}) = \sqrt{\Lambda}V\mathbf{x}$, concluimos que

$$K(\mathbf{x}, \mathbf{x}') = K_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) = \phi(\mathbf{x})^T\phi(\mathbf{x}') = \mathbf{x}^T V^T\sqrt{\Lambda}\sqrt{\Lambda}V\mathbf{x}' = \mathbf{x}^T B\mathbf{x}'$$

con lo cual ya hemos probado que es un kernel al estar en el caso anterior. □

Teorema 5.7. *Si una sucesión de kernels $K_1(\mathbf{x}, \mathbf{x}'), K_2(\mathbf{x}, \mathbf{x}'), ...$ en $\mathbb{R}^D \times \mathbb{R}^D$ tiene límite, es decir*

$$lim_{t\to\infty} K_i(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}')$$

entonces el límite $K(\mathbf{x}, \mathbf{x}')$ también es un kernel.

Demostración. Basta utilizar el teorema 5.2, como en el resto de las pruebas y considerar que cada $K_i(\mathbf{x}, \mathbf{x}')$ es un kernel; por lo tanto su matriz de Gram K_i es semidefinida positiva y el límite conserva el signo. □

Al disponer de todos estos teoremas vamos a poder definir nuevos kernels más complejos y útiles a partir de kernels sencillos. A continuación comentaremos dos kernels muy utilizados, y que se basan en los kernels básicos sobre los que se les aplica ciertas operaciones comentadas anteriormente, y que nos mantienen en la clase de kernels.

Teorema 5.8. *(Kernel polinómico) Supongamos que d es un entero positivo, entonces la función polinómica homogénea de orden d*

$$K(\mathbf{x}, \mathbf{x}') = (< \mathbf{x}, \mathbf{x}' >)^d \tag{5.9}$$

y la función polinómica no homogénea de orden d

$$K(\mathbf{x}, \mathbf{x}') = (< \mathbf{x}, \mathbf{x}' > +1)^d \tag{5.10}$$

ambas son kernels.

Demostración. La demostración es inmediata a partir de los teoremas 5.3 y 5.5. □

Teorema 5.9. *(Kernel de función de base radial Gaussiana) La función de base radial Gaussiana de parámetro σ*

$$K(\mathbf{x}, \mathbf{x}') = exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2) \tag{5.11}$$

es un kernel.

Demostración. En primer lugar, debemos de probar que si $K_1(\mathbf{x}, \mathbf{x}')$ es un kernel en $\mathbb{R}^D \times \mathbb{R}^D$ y $p(\mathbf{x})$ es un polinomio con coeficientes positivos, entonces la función $p(K_1(\mathbf{x}, \mathbf{x}'))$ es un kernel. Esto es cierto, puesto que si consideramos $p(x) = a_q x^q + ... + a_1 x + a_0$ entonces $p(K_1(\mathbf{x}, \mathbf{x}')) = a_q[K_1(\mathbf{x}, \mathbf{x}')]^q + ... + a_1 K_1(\mathbf{x}, \mathbf{x}') + a_0$
es un kernel usando los teoremas 5.4 y 5.5.

Por otra parte, probemos que si $K_1(\mathbf{x}, \mathbf{x}')$ es un kernel entonces $exp(K_1(\mathbf{x}, \mathbf{x}'))$ es un kernel, lo cual se deduce del hecho de que la función $exp(.)$ puede ser aproximada por polinomios con coeficientes positivos, y por tanto $exp(K_1(\mathbf{x}, \mathbf{x}'))$ es un límite de kernels. Por tanto, del teorema 5.7 de deducimos que es un kernel.

Por último, probemos que la función (5.11) es un kernel, para ello, puede ser descompuesta en:

$$exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2) = exp(-\|\mathbf{x}\|^2/\sigma^2) exp(-\|\mathbf{x}'\|^2/\sigma^2) exp(2 < \mathbf{x}, \mathbf{x}' > /\sigma^2)$$

de lo cual se deduce por el teorema 5.4 que es un kernel.

□

Índice de figuras

Bibliografía

[1] Andreas C.Müller y Sarah Guido (2016), *Introduction to Machine Learning with Python*. O´Reilly Media, Inc

[2] Boyd S. y Vandenbergue L. (2004), *Convex Optimization*. Cambridge University Press

[3] David C.Lay (2016) , *Álgebra lineal y sus aplicaciones*. Addison-Wesley

[4] Deisenroth, A.Aldo Faisal y Cheng Soon Ong (2020), *Mathematics for Machine Learning*. Cambridge University Press

[5] Deng N. Y. y Tian Y. J. (2004), *New methods in data mining-support vector machines*. Science Press

[6] Hamdy A. Taha (2006), *Operations Research an Introduction. Eighth edition*. Pearson Prentice Hall

[7] Naiyang Deng, Yingjie Tian y Chunhua Zhang (2013), *Support Vector Machines. Optimizacion Based Theory, Algorithms, and Extensions*. Chapman & Hall / CRC