Facultad de Ciencias
uma.es

Selección de Variables en Máquinas de Vectores Soporte.

Feature selection for support vector machine.

Trabajo Fin de Grado en Matemáticas
Universidad de Málaga

Autor: Antonio Luis Gómez Cuevas.

Área de conocimiento y/o departamento: Estadística e Investigación Operativa.

Fecha de presentación: Junio, 2024.

Tema: Aprendizaje Supervisado y Selección de Variables.

Tipo: trabajo de revisión bibliográfica.

Modalidad: individual.

Número de páginas (sin incluir introducción, bibliografía ni anexos): 58.

DECLARACIÓN DE ORIGINALIDAD DEL TFG

D./Dña. *Antonio Luis Gómez Cuevas*, con DNI (NIE o pasaporte) [redacted], estudiante del Grado en *Matemáticas* de la Facultad de Ciencias de la Universidad de Málaga, **DECLARO:**

Que he realizado el Trabajo Fin de Grado titulado "*Selección de Variables en Máquinas de Vectores Soporte.*" y que lo presento para su evaluación. Dicho trabajo es original y todas las fuentes bibliográficas utilizadas para su realización han sido debidamente citadas en el mismo.

De no cumplir con este compromiso, soy consciente de que, de acuerdo con la normativa reguladora de los procesos de evaluación de los aprendizajes del estudiantado de la Universidad de Málaga de 23 de julio de 2019, esto podrá conllevar la calificación de suspenso en la asignatura, sin perjuicio de las responsabilidades disciplinarias en las que pudiera incurrir en caso de plagio.

Para que así conste, firmo la presente en Málaga, el *04/06/2024*

Fdo:... [redacted]

Índice general

Selección de Variables en Máquinas de Vectores Soporte.

Resumen

En los últimos años, ha cobrado especial importancia la predicción de resultados basándose en la información proporcionada por datos, siendo uno de sus mayores exponentes el aprendizaje supervisado. Dos aspectos esenciales del aprendizaje supervisado son las técnicas empleadas para realizar predicciones y la selección de características. En este trabajo, se desarrollará una de las técnicas de aprendizaje supervisado más usadas y que mejor resultados proporciona para la clasificación binaria: las Máquinas de Vectores Soporte (o SVM, por sus siglas en inglés de Support Vector Machines). Para el problema de la selección de variables, se describirán tres tipos de técnicas conocidas como: *filters*, *wrappers* y *embedded*. Por otro lado, durante el desarrollo del trabajo nos centraremos tanto en los aspectos teóricos como prácticos. Finalizaremos con las conclusiones y futuras propuestas de mejora.

Palabras clave:

Aprendizaje automático, aprendizaje supervisado, clasificación binaria, SVM, selección de variables.

Feature selection for support vector machine.

Abstract

In recent years, the prediction of results based on information provided by data has become particularly important, being one of the main exponents the supervised learning. Two essential aspects of supervised learning are the techniques used to make predictions and the selection of features. In this paper, we will develop one of the most widely used and best performing supervised learning techniques for binary classification: Support Vector Machines (SVM). For the problem of variable selection, three types of techniques will be described, known as: *filters*, *wrappers* and *embedded*. On the other hand, during the development of the work we will focus on both theoretical and practical aspects. We will end with conclusions and future proposals for improvement.

key words:

MACHINE LEARNING, SUPERVISED LEARNING, BINARY CLASSIFICATION, SVM, FEATURE SELECTION.

Introducción

El aprendizaje automático o más conocido por su nombre en inglés *machine learning* es el proceso mediante el cual se utiliza la información proporcionada por los datos para construir modelos matemáticos en los que se identifican ciertos patrones. Posteriormente, estos patrones serán usados para hacer predicciones sobre elementos no vistos con anterioridad.

El aprendizaje automático utiliza distintas técnicas que podemos dividir en dos grandes grupos: aprendizaje supervisado y aprendizaje no supervisado.

Por un lado, el aprendizaje supervisado asume que se conoce un conjunto de datos observados llamado conjunto de entrenamiento o *training set*. Este conjunto está formado por los pares (x_i, y_i), $i = 1, \ldots, n$. Por un lado, cada vector $x_i \in \mathcal{R}^p$ está formado por p componentes, $x_{i1}, \ldots, x_{ip}$ que toman valores de las llamadas variables explicativas, o en inglés *features*. Es decir, $x_i = (x_{i1}, \ldots, x_{ip}) \forall i$. Nótese que, en el caso en el que nos estemos refiriendo a cada variable en general, la denotaremos por f_j, $j = 1, \ldots, p$, formando el conjunto de las variables explicativas, $P = \{f_1, \ldots, f_p\}$. Por otro lado, y_i, denota al valor de la variable respuesta de la observación i, $i = 1, \ldots, n$. De nuevo, cuando, en general, hablemos de variable respuesta, nos referiremos a y. Más adelante, hablaremos sobre el tipo de valores que puede tomar la variable respuesta. El objetivo principal del aprendizaje supervisado consiste en encontrar un modelo de predicción que relacione las variables explicativas con la variable respuesta. De esta forma, dado un nuevo dato del que solo conocemos su vector explicativo, podemos hacer una estimación del valor de la variable respuesta, usando el modelo de predicción previamente construido.

Por otro lado, en el aprendizaje no supervisado asumimos conocido un conjunto de entrenamiento que solamente está formado por los vectores explicativos. En este caso, la variable respuesta no existe. El objetivo del aprendizaje no supervisado es construir un modelo que permita encontrar patrones, y describe asociaciones entre las *features*. Dos ejemplos clásicos muy simples de aprendizaje no supervisado son el agrupamiento (conocido en inglés como *clustering*) y la reducción de la dimensionalidad (*dimensionality reduction*).

Durante todo este trabajo nos centraremos en el aprendizaje supervisado. Para más información sobre el aprendizaje no supervisado, consultar [20, 28].

Para ilustrar el concepto de aprendizaje supervisado, daremos el siguiente ejemplo:

Ejemplo 0.1. *Supongamos que tenemos un grupo de personas a las que hemos medido su altura en centímetros, y las hemos agrupado como altas o bajas. Es decir, en este caso particular la variable explicativa es la altura, mientras que la variable respuesta es alta/baja. El objetivo es crear un modelo matemático que dada la altura de una persona no estudiada anteriormente sea capaz de predecir si dicha nueva persona es alta o baja.*

Por ejemplo, podemos asumir que el conjunto de datos que hemos recogido viene dado en la Tabla 1.

variable explicativa	variable respuesta
170	alta
147	baja
150	baja
154	baja
175	alta
152	baja
173	alta
177	alta

Tabla 1: Datos del Ejemplo 0.1. En la columna izquierda los valores de la variable explicativa altura para cada observación y en la columna derecha la clase a la que pertenece.

Utilizando estos datos, podemos crear un modelo matemático para hacer la predicción de un nuevo punto. Un posible modelo de predicción podría construirse tomando como referencia para la clasificación el punto medio entre el valor de la altura más pequeño de las personas altas y el valor de la altura más grande de las personas bajas. De esta forma, nos queda que:

- *Si el valor de la variable explicativa altura $<$ 162 cm, clasificamos a la persona como baja.*
- *En caso contrario, la persona es clasificada como alta.*

Así, usando el modelo de predicción anterior, si llega una nueva persona con una altura de 165 cm, la clasificaremos como baja.

Tal y como hemos comentado anteriormente, en este trabajo nos centraremos en el aprendizaje supervisado. Nótese que en función de las características de la variable respuesta podemos distinguir dos tipos de tareas: si la variable respuesta es cuantitativa, hablaremos de regresión [28] y, por el contrario, si la respuesta es cualitativa, estaremos ante la tarea de clasificación, [28]. En particular, en este trabajo nos focalizaremos en la clasificación binaria. Es decir, trabajaremos bajo la hipótesis de que la variable respuesta del conjunto de entrenamiento, sólo puede tomar dos posibles valores. Volviendo al Ejemplo 0.1 observamos que, en ese caso, estamos ante un problema de clasificación binaria, puesto que las personas (datos) se asocian en dos grupos distintos: altas y bajas. Las etiquetas de las repuestas cualitativas (como alto/bajo) en la clasificación binaria se suelen codificar con las etiquetas $0/1$ o $-1/1$ según como este diseñado el algoritmo que se vaya a emplear.

Para conocer más información sobre la clasificación no binaria, así como la regresión, se recomienda consultar [24, 28], respectivamente.

La clasificación supervisada ha sido aplicada en diferentes ámbitos: clasificación de páginas web [2], filtrado de spam [28], en biología [53], medicina [25], agricultura [21], química [42] y comercio [17] son solo algunos ejemplos.

Existen numerosas técnicas para hacer clasificación binaria. Podemos destacar por ejemplo:

- Clasificador lineal: dado un vector de variables explicativas construimos una función dada por una combinación lineal de sus componentes. Los coeficientes que acompañan a cada componente se pueden calcular de muchas formas, pero una de las más

utilizadas es la denominada Máquinas de Vectores Soporte, o en inglés *Suport Vector Machine* (SVM), cuyos detalles se estudiarán en el Capítulo 1 de este trabajo.

- Método de k-vecinos más próximos: también conocido por su nombre en inglés k-*nearest neighbors*, $k$nn, [28]. Esta metodología utiliza las k observaciones del conjunto de entrenamiento que están más cerca de la nueva observación. La predicción de la clase de la nueva observación se determina de acuerdo a la clase más frecuente entre las k observaciones anteriormente citadas.

- Árboles de clasificación: Un árbol de clasificación, [13, 12, 24] es un clasificador definido como una serie de reglas si-entonces, o también conocidas en inglés como *if-then rules*, sobre las variables explicativas. El clasificador se representa con un árbol, [7], donde cada nodo define una partición de los datos originales que satisface las condiciones *if-then*, anteriormente mencionadas. El nodo final determina la clase de la nueva observación.

En particular, el objeto de estudio de este trabajo, como se ha mencionado anteriormente, será la técnica de clasificación conocida como Máquinas de Vectores Soporte, comúnmente abreviada como SVM, por su nombre en inglés *Support Vector Machine*. Esta metodología es una de las más conocidas y utilizadas en la literatura por su eficiencia y aplicabilidad en los problemas de la vida real. En esta técnica, la optimización matemática juega un papel crucial, destacando la programación convexa [8]. En el Capítulo 1, hablaremos con detalle sobre las Máquinas de Vectores Soporte.

El *machine learning*, y en particular, la clasificación supervisada, no solo se centra en la elección del método (SVM, árbol de clasificación, etc.) para determinar un modelo adecuado. Otro aspecto importante es la apropiada selección de características, o *feature selection*, consistente en seleccionar un subconjunto adecuado del conjunto original de características $P = \{f_1, \ldots, f_p\}$ para construir modelos más fáciles de interpretar reduciendo su complejidad, más eficientes en su elaboración y más precisos en la clasificación de datos. En el Capítulo 2 desarrollaremos tres enfoques en los que se suelen agrupar las técnicas de selección de características: métodos *filters* que son independientes del modelo seleccionado para la clasificación y solo eligen las variables en función de sus propiedades estadísticas, los métodos *wrappers* que van construyendo sucesivos modelos con las distintas características hasta seleccionar aquellas que mejor resultado ofrecen a la tarea de clasificación y los métodos *embedded* que realizan la contribución del modelo de clasificación y la selección de variables al mismo tiempo.

Capítulo 1

Máquinas de Vectores Soporte (SVM)

Las Máquinas de Vectores Soporte son una herramienta para la clasificación binaria muy conocida y usada debido a su gran utilidad y eficacia. El objetivo de este capítulo es describir la idea en la que se basa, su formulación, así como los problemas de optimización (primal y dual) necesarios para su resolución. Finalmente, se detallará el conocido como truco del kernel necesario para, la generalización del método al caso no linealmente separable.

1.1. Caso linealmente separable. SVM de margen duro.

Suponemos dado un conjunto de entrenamiento $\Omega = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, para cada observación $x_i, i = 1, \ldots, n$, los vectores predictores o explicativos viene dadas por $x_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p$. Nótese que el símbolo T indica el vector traspuesto. Si no se indica lo contrario, asumiremos que todos los vectores son columna. Además, $y_i \in \{-1, 1\}, \forall i = 1, \ldots, n$ indica el valor de la variable respuesta a la que pertenece la observación i. Dada la naturaleza del método, la variable respuesta debe ser obligatoriamente codificada como -1 y 1. Más adelante, explicaremos con detalle el porqué de esta elección. Además, nótese que, usualmente la variable respuesta viene expresada de forma cualitativa. Por ejemplo, en el caso del Ejemplo 0.1, la variable respuesta venía dada por *alto* y *bajo*. Será, por lo tanto, elección del usuario determinar qué clase se va a codificar con $+1$ y cual se establece como -1.

Supondremos, además, que los datos son linealmente separables, es decir, que existe un hiperplano lineal que separa perfectamente los datos pertenecientes a cada una de las dos clases.

Estos hiperplanos lineales se pueden describir por su ecuación normal de la forma

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

donde $w = (w_1, \ldots, w_p) \in \mathbb{R}^p$ es el vector normal del hiperplano, $b \in \mathbb{R}$ es el término independiente de la ecuación, y $\langle \cdot, \cdot \rangle$ simboliza el producto escalar.

Nótese que si los datos del conjunto de entrenamiento son linealmente separables por el hiperplano $\pi \equiv \langle w, x \rangle + b = 0$ aquellas observaciones x_i que al evaluarlos en el primer miembro de su ecuación normal cumplen $\langle w, x_i \rangle + b \geq 0$ tendrán la etiqueta $y_i = +1$. Por el contrario, los datos x_i que cumplen $\langle w, x_i \rangle + b < 0$ tienen asocia la etiqueta $y_i = -1$. Dado un conjunto de observaciones Ω, linealmente separables, es fácil apreciar que existen infinitos hiperplanos lineales que separan perfectamente ambas clases. Un ejemplo de esto puede verse en la Figura 1.1(a), donde se tiene un conjunto de datos

en $\mathbb{R}^2$ que se puede separar perfectamente usando distintos hiperplanos lineales. De esta forma, el objetivo principal del SVM es determinar, de entre todos los posibles hiperplanos aquel que maximiza el llamado margen de separación. Se puede apreciar un ejemplo de un hiperplano lineal con margen máximo en la Figura 1.1(b).

Pasaremos a definir formalmente el concepto de margen. Para ello, previamente necesitamos definir la distancia de un punto Q, a un hiperplano π:

Definición 1.1. *Dado un punto $Q = (q_1, \ldots, q_p) \in \mathbb{R}^p$ y un hiperplano π representado por su ecuación normal $\pi \equiv \langle w, x\rangle + b = 0$ con $w = (w_1, \ldots, w_p) \in \mathbb{R}^p$ el vector normal y $b \in \mathbb{R}$ el término independiente, definimos la distancia de Q a π, denotado por $D(Q, \pi)$ como:*

$$D(Q, \pi) = \frac{|\langle w, Q\rangle + b|}{\|w\|} \tag{1.1}$$

Nótese que en el caso en el que tengamos una observación de la forma $(x_i, y_i) \in \mathbb{R}^p \times \{-1, 1\}$, podemos considerar la distancia de x_i a π, y reescribir la expresión (1.1) haciendo uso de las siguientes observaciones:

- En el caso en que $y_i = 1$, tendremos que $\langle w, x_i\rangle + b \geq 0$. Por lo tanto $|\langle w, x_i\rangle + b| = \langle w, x_i\rangle + b = y_i(\langle w, x_i\rangle + b)$.
- En caso contrario, es decir cuando $y_i = -1$, tendremos que $\langle w, x_i\rangle + b < 0$. Luego $|\langle w, x_i\rangle + b| = -(\langle w, x_i\rangle + b) = y_i(\langle w, x_i\rangle + b)$.

Es decir, que en el contexto del SVM, podemos definir la distancia de un punto x_i a un hiperplano π de la siguiente forma:

Definición 1.2. *Sea x_i una observación ya clasificada, con clase y_i. Sea $\pi \equiv \langle w, x\rangle + b = 0$ un hiperplano definido por su ecuación normal con $w = (w_1, \ldots, w_p) \in \mathbb{R}^p$ el vector normal del hiperplano, $b \in \mathbb{R}$ el término independiente de la ecuación. Entonces, la distancia de x_i a π viene dada por:*

$$D(x_i, \pi) = \frac{y_i(\langle w, x_i\rangle + b)}{\|w\|} \tag{1.2}$$

Pasemos ahora a la definición de margen de un hiperplano:

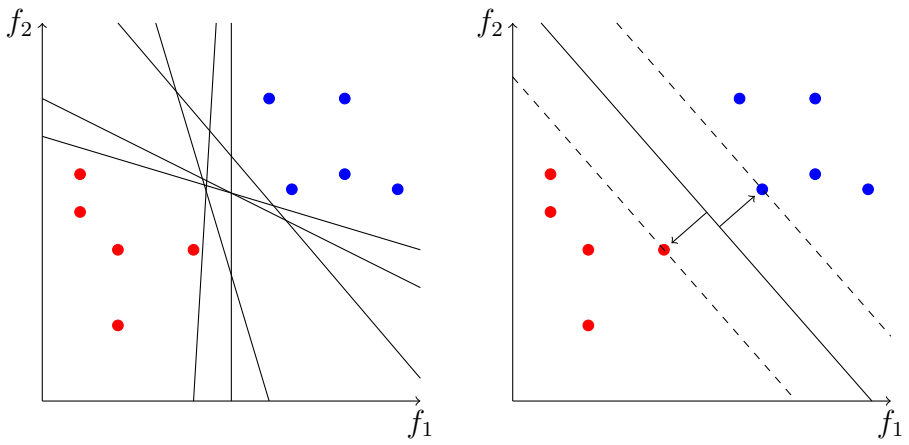
Definición 1.3. *Dado un conjunto de entrenamiento $\Omega = \{(x_1, y_1) \ldots (x_n, y_n)\}$ y un hiperplano definido por su ecuación normal $\pi \equiv \langle w, x\rangle + b = 0$ con $w = (w_1, \ldots, w_p) \in \mathbb{R}^p$ el vector normal del hiperplano, $b \in \mathbb{R}$ el término independiente de la ecuación, definimos el margen $M(w, b)$ de π respecto del conjunto de entrenamiento como:*

$$M(w, b) := \text{mín}\{D(x_i, \pi) : i = 1, \ldots, n\}$$

donde $D(\pi, x_i)$ es la distancia del dato x_i del conjunto de entrenamiento al hiperplano π, dada por (1.2).

Por tanto, para encontrar el hiperplano óptimo, es decir, los valores de w y b óptimos debemos maximizar el margen $M(w, b)$. Esto nos lleva a resolver el siguiente problema de optimización:

$$\begin{aligned} \max_{w,b} \quad & M(w, b) \equiv \max_{w,b} \left(\min_{i=1,\ldots,n} \frac{y_i(\langle w, x_i\rangle + b)}{\|w\|} \right) \\ \text{s.a.} \quad & y_i(\langle w, x_i\rangle + b) \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{1.3}$$



(a) Hiperplanos separadores (b) Hiperplano de margen máximo

Figura 1.1: En la imagen a la izquierda tenemos varios hiperplanos separadores. Mientras en la imagen de la derecha el de margen óptimo generado por SVM.

donde la restricción en (1.3) indica que todas las observaciones se clasifican correctamente. Esto es debido a que los puntos con $\langle w, x_i\rangle + b \geq 0$ se clasifican en la clase $+1$ y las que cumplen $\langle w, x_i\rangle + b < 0$ se clasifican en la clase -1. Además, dado que el denominador de la expresión en la función objetivo no depende del índice i, podemos reescribir el problema de optimización como:

$$\begin{aligned} \max_{w,b} \quad & \left(\frac{1}{\|w\|} \min_{i=1,\dots,n} y_i(\langle w, x_i\rangle + b)\right) \\ \text{s.a.} \quad & y_i(\langle w, x_i\rangle + b) \geq 0, \quad i = 1, \dots, n \end{aligned} \tag{1.4}$$

Obsérvese que, si utilizamos esta formulación, se tiene que si (w^*, b^*) es solución óptima de (1.4), entonces $(\lambda w^*, \lambda b^*)$ también es solución óptima de (1.4), $\forall \lambda > 0$, puesto que se satisface la restricción de factibilidad: $y_i(\langle \lambda w, x_i\rangle + \lambda b) = \lambda y_i(\langle w, x_i\rangle + b) \geq 0$ y el valor objetivo no cambia, tal y como podemos comprobar:

$$\begin{aligned} \frac{1}{\|\lambda w\|} \min_{i=1,\dots,n} y_i(\langle \lambda w, x_i\rangle + \lambda b) &= \frac{1}{\lambda\|w\|} \min_{i=1,\dots,n} \lambda y_i(\langle w, x_i\rangle + b) = \\ &= \frac{\lambda}{\lambda\|w\|} \min_{i=1,\dots,n} y_i(\langle w, x_i\rangle + b) = \frac{1}{\|w\|} \min_{i=1,\dots,n} y_i(\langle w, x_i\rangle + b) \end{aligned}$$

Es decir, que si existe solución, entonces existen infinitas soluciones óptimas. Todas estas soluciones tendrán la misma dirección pero distinta norma. Dado que estamos interesados en la dirección del vector normal del hiperplano, y no en su norma, parece razonable buscar unos valores de (w, b) tales que:

$$\min_{i=1,\dots,n} y_i(\langle w, x_i\rangle + b) = 1$$

o equivalentemente

$$M(w, b)\|w\| = 1$$

por la siguiente sucesión de equivalencias:

$$M(w, b)\|w\| = 1 \Leftrightarrow \left(\min_{i=1,\dots,n} \frac{y_i(\langle w, x_i\rangle + b)}{\|w\|}\right)\|w\| = 1 \Leftrightarrow \min_{i=1,\dots,n} y_i(\langle w, x_i\rangle + b) = 1$$

De esta forma, eliminamos el problema de la multiplicidad de soluciones, [11]. Además de la condición $M(w,b)\|w\| = 1$, obtenemos una expresión del margen que solo depende de w:

$$M(w,b) = \frac{1}{\|w\|}$$

Dado que hemos impuesto que $\min_{i=1,\dots,n} y_i(\langle w, x_i\rangle + b) = 1$, entonces se tiene que:

$$y_i(\langle w, x_i\rangle + b) \geq 1, i = 1, \dots, n \tag{1.5}$$

Luego el problema de optimización (1.4) queda reescrito como:

$$\begin{aligned} \max_{w,b} \quad & \frac{1}{\|w\|} \\ \text{s.a.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1, \quad i = 1, \dots, n \end{aligned} \tag{1.6}$$

Dado que maximizar una función cualquiera $f(x)$ es equivalente a minimizar su valor inverso $\frac{1}{f(x)}$, tenemos que el Problema (1.6) se puede reescribir como:

$$\begin{aligned} \min_{w,b} \quad & \|w\| \\ \text{s.a.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1, \quad i = 1, \dots, n \end{aligned} \tag{1.7}$$

El Problema (1.7) se trata de un problema de optimización convexo [8]. Utilizando las propiedades de la programación matemática, podemos sustituir la función objetivo de (1.7), por $\Psi(\|w\|)$, donde Ψ es cualquier función creciente en $\mathbb{R}^+$, [13]. En nuestro caso, $\Psi(t) = \frac{1}{2}t^2$, llegando a la formulación que es la más usada en la literatura y equivalente a (1.7):

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.a.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1, \quad i = 1, \dots, n \end{aligned} \tag{1.8}$$

El problema de optimización (1.8) es un problema de optimización convexo por tener que minimizar una función cuadrática con coeficientes positivos sujeta a restricciones lineales. La formulación (1.8) se conoce bajo el nombre de SVM de margen duro (*hard-margin SVM*).

Aunque para resolver el problema (1.8) puede usarse las técnicas de programación convexa [8], normalmente se suele resolver una formulación alternativa, pero equivalente, conocida como formulación dual.

A continuación, mostraremos como obtener la formulación dual del problema de margen duro. Para ello, necesitamos algunos conceptos y resultados, empezando por un teorema que nos proporciona las condiciones necesarias para que la solución de un problema de programación matemática sea óptima:

Teorema 1.1 (Condiciones Karush-Kuhn-Tucker, KKT)**.** *Dado un problema de optimización primal*

$$\begin{aligned} \min_{w} \quad & f(w) \\ s.a \quad & g_i(w) \geq 0 \quad i = 1, \dots, m \\ & h_i(w) = 0 \quad i = 1, \dots, t \end{aligned} \tag{1.9}$$

donde las funciones f, g_i y h_i son diferenciables de $\mathbb{R}^p$ en $\mathbb{R}$. Si el punto $w^* \in \mathbb{R}^p$ es una solución óptima del problema, entonces w^* verifica las siguientes condiciones:

$$\nabla f(w^*) - \sum_{i=1}^{m} \alpha_i \nabla g_i(w^*) + \sum_{i=1}^{t} \nu_i \nabla h_i(w^*) = 0 \tag{1.10}$$

$$g_i(w^*) \geq 0 \qquad i = 1, \ldots, m \tag{1.11}$$

$$h_i(w^*) = 0 \qquad i = 1, \ldots, t \tag{1.12}$$

$$\alpha_i \geq 0 \qquad i = 1, \ldots, m \tag{1.13}$$

$$\alpha_i g_i(w^*) = 0 \qquad i = 1, \ldots, m \tag{1.14}$$

para algunos vectores $\alpha \in \mathbb{R}^m$ y $\nu \in \mathbb{R}^t$, y siendo $\nabla\phi$ el gradiente de una función ϕ de $\mathbb{R}^p$,

$$\nabla\phi = \left(\frac{\partial\phi}{\partial w_1}, \ldots, \frac{\partial\phi}{\partial w_p} \right)$$

Estas condiciones se conocen con el nombre de condiciones KKT debido a los autores que los enunciaron por primera vez, Karush, Kuhn y Tucker.

Es importante destacar que el caso de un problema de optimización convexa, estas condiciones son también suficientes, [8].

Dado que queremos hallar la formulación dual del Problema (1.8) vamos a definir el concepto de Dual de Wolfe [35, 49].

Definición 1.4 (Dual de Wolfe)**.** *Dado el problema de optimización primal*

$$\begin{aligned} \min_{w} \quad & f(w) \\ s.a \quad & g_i(w) \geq 0 \quad i = 1, \ldots, m \end{aligned} \tag{1.15}$$

para una función f convexa y diferenciable, y $g_i, i = 1, \ldots, m$ funciones cóncavas y diferenciables, todas ellas definidas de $\mathbb{R}^p$ en $\mathbb{R}$, definimos el problema dual de Wolfe como:

$$\begin{aligned} \max_{w,\alpha} \quad & L(w, \alpha) \equiv f(w) + \sum_{i=1}^{m} \alpha_i g_i(w) \\ s.a \quad & \frac{\partial L(w, \alpha)}{\partial w} \equiv \nabla f(w) - \sum_{i=1}^{m} \alpha_i \nabla g_i(w) = 0 \\ & \alpha_i \geq 0, \quad i = 1, \ldots, m \end{aligned} \tag{1.16}$$

donde las variables α_i para $i = 1, \ldots, n$ se llaman variables duales, a la función $L(w, \alpha)$ se la conoce por función lagrangiana o lagrangiano.

Obsérvese que las restricciones del dual de Wolfe (1.16) coinciden, respectivamente, con las condiciones KKT (1.10) y (1.13), considerando que el problema (1.9) puede verse como (1.15) tomando h_i la función constante igual a cero, $\forall i = 1, \ldots, t$.

El siguiente teorema relaciona la solución óptima del problema primal con la del dual de Wolfe.

Teorema 1.2. *Sea (w^*, α^*) solución del problema dual (1.16) donde $f(w)$ es convexa y dos veces continuamente diferenciable y las funciones de restricción $g_i(w)$ son cóncavas y dos veces continuamente diferenciables. Entonces, se tiene que w^* es solución óptima*

de (1.15) y $f(w^*) = L(w^*, \alpha^*)$ si $f(w)$ es estrictamente convexa en el entorno de w^* o al menos una de las restricciones del primal $g_i(w)$ activas (es decir, aquellas para las que $\alpha^* \neq 0$) es estrictamente cóncava en el entorno de w^*. Si $f(w)$ es cuadrática y las restricciones son lineales, este teorema es cierto si $f(w)$ es sólo convexa y dos veces continuamente diferenciable.

Se puede encontrar una demostración de este teorema en [35].

Utilizando la Definición 1.4, calculamos el modelo dual del problema SVM de margen duro (1.8). Para ello, primero identificamos las restricciones g_i del dual de Wolfe con las restricciones que hay en (1.8), a saber:

$$g_i(w, b) = y_i(\langle w, x_i\rangle + b) - 1, \quad i = 1, \ldots, n$$

Por otro lado, las variables duales asociadas a las restricciones g_i las denotaremos por α_i, $i = 1, \ldots, n$.

Para obtener el problema dual de (1.8) comenzamos calculando su función lagrangiana, que viene dado por:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i[y_i(\langle w, x_i\rangle + b) - 1] \tag{1.17}$$

la cual maximizaremos respecto de w y b. Para obtener la restricción del dual de Wolfe, derivamos con respecto de todas las variables de decisión del primal e igualamos a cero:

$$\frac{\partial L(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{1.18}$$

$$\frac{\partial L(w, b, \alpha)}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.19}$$

Llegamos al problema dual de Wolfe para (1.8):

$$\begin{aligned} \max_{w,b,\alpha} \quad & \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i[y_i(\langle w, x_i\rangle + b) - 1] \\ \text{s.a.} \quad & w = \sum_{i=1}^{n} \alpha_i y_i x_i \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{1.20}$$

Usando (1.18) y (1.19) en (1.17) obtenemos la función objetivo del Problema (1.20) de forma más simplificada.

Para ello, primero desarrollamos de forma conveniente el sumatorio de la función lagrangiana en otros tres términos, resultando:

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

A continuación, observamos que el término $-b\sum_{i=1}^{n}\alpha_i y_i$ desaparece por la Ecuación (1.19), quedando:

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

Finalmente, empleamos la Ecuación (1.18), la bilinealidad y la simetría del producto escalar para simplificar los dos primeros términos:

$$\begin{aligned} L(w,b,\alpha) =& \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i y_i\langle w, x_i\rangle + \sum_{i=1}^{n}\alpha_i \overset{(1.18)}{=} \\ & \frac{1}{2}\left\langle \sum_{i=1}^{n}\alpha_i y_i x_i, \sum_{\ell=1}^{n}\alpha_\ell y_\ell x_\ell \right\rangle - \sum_{i=1}^{n}\alpha_i y_i \left\langle \sum_{\ell=1}^{n}\alpha_\ell y_\ell x_\ell, x_i \right\rangle + \sum_{i=1}^{n}\alpha_i = \\ & \frac{1}{2}\sum_{i=1}^{n}\sum_{\ell=1}^{n}\alpha_i y_i \alpha_\ell y_\ell \langle x_i, x_\ell\rangle - \sum_{i=1}^{n}\sum_{\ell=1}^{n}\alpha_i y_i \alpha_\ell y_\ell \langle x_\ell, x_i\rangle + \sum_{i=1}^{n}\alpha_i = \\ & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{\ell=1}^{n}\alpha_i \alpha_\ell y_i y_\ell \langle x_i, x_\ell\rangle \end{aligned}$$

Dando como resultado final la siguiente formulación de la función objetivo del dual:

$$L(w,b,\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{\ell=1}^{n}\alpha_i \alpha_\ell y_i y_\ell \langle x_i, x_\ell\rangle$$

Esta función objetivo estará condicionada a las restricciones (1.19) y $\alpha_i \geq 0$, $i = 1, \ldots, n$.

Como consecuencia, podemos formular el problema de optimización dual del SVM de margen duro (1.8) que viene dado por:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{\ell=1}^{n}\alpha_i \alpha_\ell y_i y_\ell \langle x_i, x_\ell\rangle \\ \text{s.a.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \quad i = 1, \ldots, n \\ & \alpha_i \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{1.21}$$

Nótese que el Problema (1.21) sólo depende de las variables duales α. El Problema (1.21) es un problema de programación convexa con función objetivo cuadrática y restricciones lineales que se pueden resolver usando las técnicas de [8].

Una vez calculada la solución óptima α^* de (1.21), para hallar la solución óptima del problema primal (1.8), es decir, (w^*, b^*), usamos la Ecuación (1.18) y la condición KKT (1.14). Partiendo de (1.18) obtenemos directamente la expresión para w^*:

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

Por tanto, w^* sólo se expresa en función de las observaciones con $\alpha_i^* \neq 0$. A dichas observaciones se les llama vectores soporte.

Veamos ahora cómo se obtiene b^*. Para ello, haremos uso de la condición (1.14) aplicada a nuestro problema, donde $g_i(w,b) = y_i(\langle w, x_i\rangle + b) - 1$, para $i = 1, \ldots, n$.

Tenemos, por tanto,

$$\alpha_i^*[y_i(\langle w^*, x_i\rangle + b^*) - 1] = 0, \quad i = 1, \ldots, n$$

Es decir, que si existiese un vector soporte $i_0 \in \{1, \ldots, n\}$ tal que $\alpha_{i_0} \neq 0$, se tendría que:

$$y_{i_0} b^* = 1 - y_{i_0}\langle w^*, x_{i_0}\rangle \tag{1.22}$$

Multiplicando (1.22) por y_{i_0}, y, teniendo en cuenta que $y_{i_0}^2 = 1$, independientemente de que la variable respuesta valga $+1$ o -1, tenemos:

$$b^* = y_{i_0} - \langle w^*, x_{i_0}\rangle \tag{1.23}$$

En la práctica, existen muchas observaciones que pueden ser vectores soporte. Es por ello, que en lugar de (1.23) suele usarse la siguiente expresión:

$$b^* = \frac{1}{|vs|}\sum_{i \in vs} y_i - \langle w^*, x_i\rangle \tag{1.24}$$

donde vs denota el conjunto de todos los vectores soporte. Es decir, se suele usar la media de los valores obtenidos en la expresión (1.23) para todos los vectores soporte.

En el razonamiento anterior, hemos comenzado asumiendo que existen vectores soporte. En la siguiente proposición, se mostrará que esta hipótesis es cierta.

Proposición 1.1. *Sea $\Omega = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ un conjunto de entrenamiento linealmente separable y sea $\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*)$ solución óptima de (1.21) para dicho conjunto de entrenamiento, entonces existe $i_0 \in \{1, \ldots, n\}$ tal que $\alpha_{i_0}^* \neq 0$.*

Demostración. Supongamos por reducción al absurdo que $\alpha_i^* = 0$ para todo $i = 1, \ldots, n$ y sea (w^*, b^*) la solución óptima de (1.8), es decir que, el mínimo de la función $f(w) = \frac{1}{2}\|w\|^2$ se alcanza en $w^* = 0$ y se cumple (1.5) para todo $i = 1, \ldots, n$. Por otro lado, por la Ecuación (1.18) tenemos que $w^* = 0$, lo que trasforma a la restricción (1.5), que coincide con la condición KKT (1.11), para $g_i(w,b) = y_i(\langle w, x_i\rangle + b) - 1$, en $y_i b^* \geq 1$ para todo $i = 1, \ldots, n$, es decir, para valores de i tal que $y_i = 1$ tenemos que $b^* \geq 1$, pero para valores de i tal que $y_i = -1$ llegamos a $b^* \leq -1$. Estas dos situaciones no pueden darse a la vez para un valor único de b^*. Es decir, hemos llegado a una contradicción. La hipótesis inicial es falsa, y, por lo tanto, existe i_0 tal que $\alpha_{i_0} \neq 0$. □

Luego, una vez, que hemos determinado la solución óptima (w^*, b^*), sólo quedaría ver cómo es la predicción de un nuevo punto x^{test}. Tenemos:

- $\widehat{Y}(x^{test}) = +1$, si $\langle w^*, x^{test}\rangle + b^* \geq 0$
- $\widehat{Y}(x^{test}) = -1$, si $\langle w^*, x^{test}\rangle + b^* < 0$

1.2. SVM de margen blando

En la sección anterior, hemos asumido que los datos del conjunto de entrenamiento son perfectamente separables por un hiperplano lineal. Sin embargo, hay muchas ocasiones en la que esto no es posible. Es decir, que, usando un separador lineal, hay algunas observaciones que se han clasificado en la clase errónea. A modo ilustrativo, en la Figura 1.2 aparecen varios hiperplanos lineales que son incapaces de separar perfectamente ambas clases.

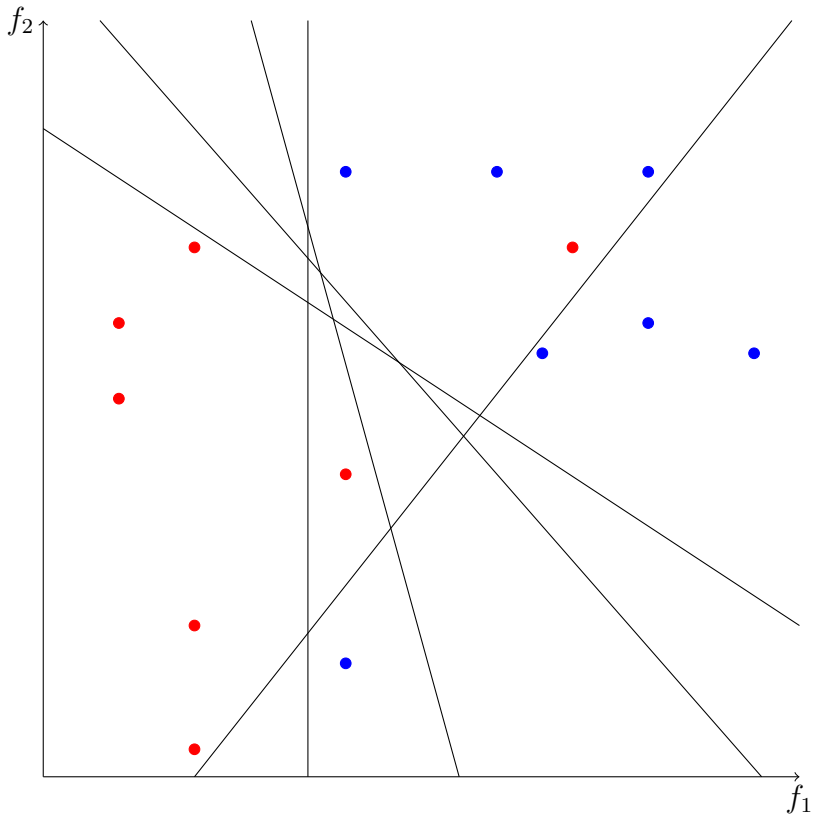


Figura 1.2: Conjunto de datos en $\mathbb{R}^2$ imposibles de separar mediante hiperplano lineal.

Una forma de lidiar con este problema consiste en maximizar el margen $M(w, b)$, que detallamos en la Definición 1.3, pero permitiendo que algunos puntos estén en el lado equivocado de dicho margen. Para ello, introducimos unas variables de holgura $\xi_i \geq 0$, $\forall i = 1, \ldots, n$, una por cada una de las observaciones.

La idea es que las variables de holgura midan la desviación desde el punto mal clasificado hasta el borde del margen de su clase. Los puntos con $\xi_i \geq 1$ son aquellos datos que se clasifican incorrectamente. Por otro lado, las observaciones con $0 < \xi_i < 1$ se clasifican correctamente, pero se sitúan en el lado equivocado del margen. Finalmente, las observaciones con $\xi_i = 0$ son aquellas que además de estar bien clasificadas, están en el lado del margen que le corresponde a su clase. Todas estas situaciones se pueden contemplar en la Figura 1.3.

Desde el punto de vista del problema de optimización, adaptamos el modelo de margen duro (1.8) modificando las restricciones de la siguiente forma:

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

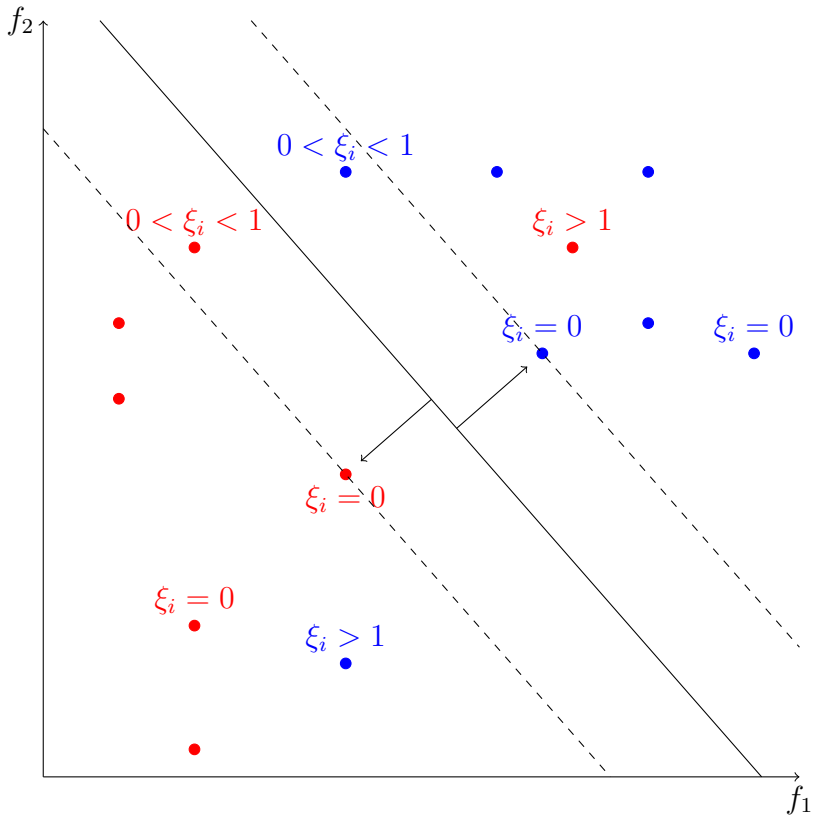


Figura 1.3: SVM de margen blando donde vemos la interpretación de los valores de ξ_i. Las observaciones con $\xi_i = 0$ son las que están bien clasificadas y pertenecen al lado del margen correcto, aquellas con $0 < \xi_i < 1$ están bien clasificadas, pero en el lado equivocado del margen y las que tienen $\xi_i \geq 1$ están las mal clasificadas.

Por otro lado, en la función objetivo además de minimizar la norma de w, $\frac{1}{2}\|w\|^2$, también queremos que los errores cometidos no sean excesivamente grandes. Es por ello que añadimos el término $\sum_{i=1}^{n} \xi_i$ penalizado con un parámetro $C > 0$. De esta forma, el problema de optimización resultante, denominado SVM de margen blando (*soft-margin SVM*), quedaría:

$$\begin{aligned} \min_{w,b,\xi_i} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.a.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{1.25}$$

El parámetro de regularización, C, es elegido por el usuario y controla el equilibrio entre un margen amplio y el número de puntos mal clasificados. Valores grandes de C, construirán hiperplanos con márgenes pequeños, pero clasificarán bien la mayoría de los puntos. Por otro lado, valores pequeños de C implicará que los márgenes sean más grandes, incluso si clasifica erróneamente más datos de entrenamiento. El Problema (1.25) se conoce como problema SVM de margen blando.

Finalmente, una vez calculada una solución óptima w y b de (1.25), dada por w^* y b^*, respectivamente, la regla de clasificación lineal para un nuevo punto x^{test} es la misma que la descrita en la sección anterior mediante la función $\widehat{Y}(x^{test}) = sign(\langle w^*, x^{test}\rangle + b^*)$.

Al igual que ocurría anteriormente, el problema SVM de margen blando dado por

(1.25) es un problema de optimización convexa, [8], que se puede resolver tanto con su fórmula primal (Problema (1.25)) como dual, dando soluciones óptimas equivalentes.

Utilizando la Definición 1.4, calculamos el modelo dual de (1.25) de forma análoga al caso de margen duro. Para este caso identificamos las restricciones g_i del dual de Wolfe con los dos tipos de restricciones que hay en (1.25), a saber:

$$g_i^1(w,b,\xi_i) = y_i(\langle w, x_i\rangle + b) - (1-\xi_i), \qquad i = 1,\dots,n \tag{1.26}$$

$$g_i^2(w,b,\xi_i) = \xi_i, \qquad i = 1,\dots,n \tag{1.27}$$

Por otro lado, las variables duales asociadas a las restricciones g_i^1 las denotaremos por α_i, $i = 1,\dots,n$, mientras que las variables duales de g_i^2 se denotaran por μ_i, $i = 1,\dots,n$.

Para obtener el problema dual de (1.25) empezamos calculando su función lagrangiana, que viene dado por:

$$L(w,b,\xi,\alpha,\mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\langle w, x_i\rangle + b) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i \tag{1.28}$$

que maximizaremos respecto de las variables del primal w, b, ξ_i. Obtenemos la restricción del dual de Wolfe. Derivamos la función lagrangiana respecto de todas las variables de decisión del primal e igualamos a cero: Para ello,

$$\frac{\partial L(w,b,\xi,\alpha,\mu)}{\partial w} = w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n}\alpha_i y_i x_i \tag{1.29}$$

$$\frac{\partial L(w,b,\xi,\alpha,\mu)}{\partial b} = \sum_{i=1}^{n}\alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{1.30}$$

$$\frac{\partial L(w,b,\xi,\alpha,\mu)}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad \Rightarrow \quad \alpha_i = C - \mu_i, \forall i = 1,\dots,n \tag{1.31}$$

Llegamos al problema dual de Wolfe para (1.25):

$$\begin{aligned} \max_{w,b,\xi,\alpha,\mu} \ & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\langle w, x_i\rangle + b) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i \\ \text{s.a.} \quad & w = \sum_{i=1}^{n}\alpha_i y_i x_i \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & \alpha_i = C - \mu_i, \quad i = 1,\dots,n \\ & \alpha_i \geq 0, \quad i = 1,\dots,n \\ & \mu_i \geq 0, \quad i = 1,\dots,n \end{aligned} \tag{1.32}$$

El Problema (1.32) se simplificará usando (1.29)-(1.31) en (1.28).

Para ello, primero aplicamos la propiedad distributiva en el primer sumatorio y desarrollamos de forma conveniente el segundo sumatorio de la función lagrangiana en otros cuatro términos, resultando:

$$\begin{aligned} L(w,b,\xi,\alpha,\mu) =& \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\langle w, x_i\rangle + b) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i = \\ & \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}C\xi_i - \sum_{i=1}^{n}\alpha_i y_i\langle w, x_i\rangle - b\sum_{i=1}^{n}\alpha_i y_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\mu_i\xi_i \end{aligned}$$

A continuación, se agrupan el primer y último sumatorio y sustituimos los valores de $C - \mu_i$ por α_i, $\forall i$ siguiendo (1.31):

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

La última igualdad se tiene, ya que se cancelan los términos iguales de signo contrario.

A continuación, observamos que el término $-b\sum_{i=1}^{n} \alpha_i y_i$ desaparece por la Ecuación (1.30), quedando:

$$\begin{aligned} L(w,b,\xi,\alpha,\mu) =& \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i \langle w, x_i\rangle - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i = \\ & \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i \langle w, x_i\rangle + \sum_{i=1}^{n} \alpha_i \end{aligned}$$

Finalmente, empleamos la Ecuación (1.29), la bilinealidad y la simetría del producto escalar para simplificar los dos primeros términos:

$$\begin{aligned} L(w,b,\xi,\alpha,\mu) =& \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i \langle w, x_i\rangle + \sum_{i=1}^{n} \alpha_i \overset{(1.29)}{=} \\ & \frac{1}{2}\left\langle \sum_{i=1}^{n} \alpha_i y_i x_i, \sum_{\ell=1}^{n} \alpha_\ell y_\ell x_\ell \right\rangle - \sum_{i=1}^{n} \alpha_i y_i \left\langle \sum_{\ell=1}^{n} \alpha_\ell y_\ell x_\ell, x_i \right\rangle + \sum_{i=1}^{n} \alpha_i = \\ & \frac{1}{2}\sum_{i=1}^{n}\sum_{\ell=1}^{n} \alpha_i y_i \alpha_\ell y_\ell \langle x_i, x_\ell\rangle - \sum_{i=1}^{n}\sum_{\ell=1}^{n} \alpha_i y_i \alpha_\ell y_\ell \langle x_\ell, x_i\rangle + \sum_{i=1}^{n} \alpha_i = \\ & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell \langle x_i, x_\ell\rangle \end{aligned}$$

Dando como resultado final la siguiente formulación de la función objetivo del dual:

$$L(w,b,\xi,\alpha,\mu) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell \langle x_i, x_\ell\rangle$$

Esta función objetivo estará condicionada a las restricciones (1.30) y

$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

Nótese que esta última restricción se obtiene porque sabemos que $\alpha_i \geq 0$ y $\mu_i \geq 0$, $\forall i$. Además, por (1.31), tenemos que $\mu_i = C - \alpha_i$. Luego, $C - \alpha_i \geq 0$, $\forall i$, o lo que es equivalente, $\alpha_i \leq C$, $\forall i$.

Como consecuencia, podemos formular el problema de optimización dual del SVM de margen blando (1.25) que viene dado por:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell \langle x_i, x_\ell \rangle \\ \text{s.a.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \quad i = 1, \ldots, n \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \end{aligned} \tag{1.33}$$

El problema de optimización (1.33) se trata de un problema de optimización convexa, donde se maximiza una función objetivo cóncava sujeta a restricciones lineales, y que puede resolverse usando las estrategias de [8].

Obsérvese que, al igual que en el problema dual de margen duro (1.21), la solución óptima de (1.33) dada por α^*, nos permitirá obtener la solución óptima del primal, w^* de forma sencilla, simplemente aplicando la condición (1.29), resultando que

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

Para calcular el valor de b^* será primero necesario probar que, en efecto, existen vectores soporte. Para ello se usarán las ecuaciones (1.29)-(1.31), además de las condiciones de Karush-Kuhn-Tucker del Problema (1.25):

$$\alpha_i [y_i(\langle w, x_i \rangle + b) - (1 - \xi_i)] = 0 \tag{1.34}$$

$$\mu_i \xi_i = 0 \tag{1.35}$$

$$y_i(\langle w, x_i \rangle + b) - (1 - \xi_i) \geq 0 \tag{1.36}$$

para $i = 1, \ldots, n$.

El siguiente resultado nos da una demostración de la existencia de vectores soporte.

Proposición 1.2. *Sea $\Omega = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ un conjunto de entrenamiento y sea $\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*)$ solución óptima de (1.33) para dicho conjunto de entrenamiento, entonces existe $i_0 \in \{1, \ldots, n\}$ tal que $\alpha_{i_0}^* \neq 0$.*

Demostración. Supongamos por reducción al absurdo que $\alpha_i^* = 0$ para todo $i = 1, \ldots, n$ y sea (w^*, b^*, ξ^*) la solución óptima de (1.25).

Denotamos por μ_i^* a la variable dual óptima asociada a la restricción (1.27). Por un lado, si $\alpha_i^* = 0$ para todo $i = 1, \ldots, n$, de la Ecuación (1.31) tenemos que $\mu_i^* = C$ para todo $i = 1, \ldots, n$ y en consecuencia por (1.35) tenemos que $\xi_i^* = 0$ para todo $i = 1, \ldots, n$. Por otro lado, por la Ecuación (1.29) tenemos que $w^* = 0$. Teniendo en cuenta que $w^* = 0$ y que $\xi_i^* = 0$ para todo $i = 1, \ldots, n$, la condición KKT (1.36) se trasforma en $y_i b^* \geq 1$ para todo $i = 1, \ldots, n$. Es decir, para valores de i tal que $y_i = 1$ tenemos que $b^* \geq 1$, pero para valores de i tal que $y_i = -1$ llegamos a $b^* \leq -1$. Estas dos situaciones no pueden darse a la vez para un valor único de b^*. Dicho de otra forma, hemos llegado a una contradicción. La hipótesis inicial es falsa, y, por lo tanto, existe i_0 tal que $\alpha_{i_0}^* \neq 0$. □

En la demostración anterior, hemos probado que existen los vectores soporte. Es decir, que existen al menos una observación i tal que $\alpha_i^* \neq 0$, o equivalentemente $0 < \alpha_i^* \leq C$.

De la demostración también concluimos que los puntos con $\alpha_i^* = 0$ tienen asociados una variable de holgura nula, es decir, $\xi_i^* = 0$. Por lo tanto, esta observación i está bien clasificada y pertenece al lado correcto del margen, tal y como se aprecia en la Figura1.3.

A continuación, analizaremos la relación entre las observaciones i tal que $0 < \alpha_i \leq C$ y ξ_i:

Comenzamos con el caso en el que existe algún i tal que $0 < \alpha_i < C$. Con esta información, y usando la Expresión (1.31), tenemos que $0 < \mu_i < C$. Aplicando esta a (1.35). Es decir, que aquellos elementos con $0 < \alpha < C$ verifican que $\xi_i = 0$. Por lo tanto, aplicando (1.34) concluimos que $y_i(\langle w, x_i\rangle + b) = 1$, o lo que es lo mismo, la observación está bien clasificada y se encuentra justo encima del margen.

Pasaremos a estudiar como calcular el valor de b^* óptimo. Comenzaremos asumiendo que tenemos un vector i tal que $0 < \alpha < C$.

En este caso, y de forma análoga a como se concluía en (1.23), tenemos que $y_i(\langle w^*, x_i\rangle + b) = 1$, o lo que es lo mismo:

$$b^* = 1 - \langle w^*, x_i\rangle \tag{1.37}$$

Para finalizar, analizaremos el caso degenerado en el que no existe ninguna observación i tal que $0 < \alpha_i^* < C$, o lo que es lo mismo que $\alpha_i^* = C, \forall i$. En esta situación, para calcular b^*, Primero obtendremos el valor de w^* a partir de (1.29). Luego, por (1.34), observamos que $y_i(\langle w^*, x_i\rangle + b) - (1 - \xi_i^*) = 0$, o lo que es lo mismo $\xi_i^* = 1 - y_i(\langle w^*, x_i\rangle + b)$.

Es decir, que la formulación original (1.25) se puede escribir como:

$$\begin{aligned} \min_b \quad & \frac{1}{2}\|w^*\|^2 + C\sum_{i=1}^{n} 1 - y_i(\langle w^*, x_i\rangle + b) \\ \text{s.a.} \quad & y_i(\langle w^*, x_i\rangle + b) \geq 1 - 1 + y_i(\langle w^*, x_i\rangle + b), \quad i = 1, \ldots, n \\ & 1 - y_i(\langle w^*, x_i\rangle + b) \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{1.38}$$

El Problema (1.38) es un problema lineal con una única variable de decisión, que se puede resolver usando por ejemplo el algoritmo del símplex [38].

La primera restricción puede eliminarse, ya que no aporta ninguna información. Por otro lado, la segunda restricción también puede suprimirse e incorporarla a la función objetivo a través de la función $(\cdot)_+$ que para un valor cualquiera $a \in \mathbb{R}$, devuelve el máximo entre su valor y cero. Es decir, $(a)_+ = \max(a, 0)$. Por ello, y dado que se debe cumplir que $1 - y_i(\langle w^*, x_i\rangle + b) \geq 0$, $\forall i$, podemos reescribir (1.38) como:

$$\min_b \frac{1}{2}\|w^*\|^2 + C\sum_{i=1}^{n}(1 - y_i(\langle w^*, x_i\rangle + b))_+ \tag{1.39}$$

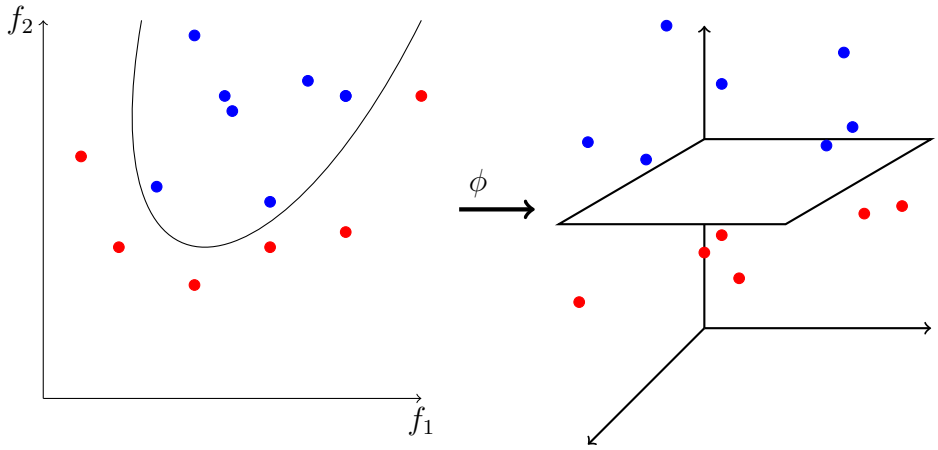
De esta forma, para calcular b^* bastaría con resolver (1.39). Para más información se recomienda consultar [10, 13].

1.3. Caso no linealmente separable

Hasta ahora sólo hemos estudiado bases de datos que se pueden separar a través de un hiperplano lineal. Sin embargo, existen situaciones en las que este tipo de funciones no es suficiente. Por ejemplo, podemos encontrarnos con bases de datos cuya separación debe hacerse a través de una función no lineal. Como ilustración, podemos observar la Figura 1.4(a) donde se aprecia un conjunto de datos que podemos separar perfectamente a través de una función que no es lineal.

Por todo esto, se hace imprescindible generalizar el modelo SVM visto en las Secciones 1.1 y 1.2. Para ello, consideramos una función $\phi : \mathbb{R}^p \to \mathcal{F}$ que traslada los puntos

x_i $\forall i = 1, \ldots, n$ a un espacio $\mathcal{F}$ de mayor dimensión, donde los datos son linealmente separables como podría ser el ejemplo de la Figura 1.4(b). A la función ϕ se la conoce por su nombre en inglés *feature map*, mientras que el espacio $\mathcal{F}$ es llamado espacio de características (o en inglés *feature space*).



(a) Conjunto separado por función no lineal (b) Conjunto trasformado linealmente separable

Figura 1.4: Imagen de un ejemplo ilustrativo, en la que vemos como la función ϕ traslada las observaciones del espacio inicial (*input space*) $\mathbb{R}^2$ donde no son separables linealmente, a un espacio de características (*feature space*) $\mathcal{F}$ donde sí son separables.

De esta forma, el conjunto de datos original $\Omega = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ se ha trasformado en $\{(\phi(x_1), y_1), \ldots, (\phi(x_n), y_n)\}$. A continuación, procedemos a construir el modelo SVM adaptándolo al caso no lineal. Nótese que, tal y como ocurría en el caso lineal, podemos considerar tanto la versión del margen duro como la del margen blando. No obstante, con objeto de evitar repeticiones, en este caso nos centraremos solo en el modelo SVM de margen blando para el caso no lineal, puesto que la metodología del margen duro no es más que un caso particular de esta, que se obtiene tomando $C = +\infty$.

Por lo tanto, partiendo del conjunto de datos $\{(\phi(x_1), y_1), \ldots, (\phi(x_n), y_n)\}$, buscamos un hiperplano lineal en el nuevo espacio $\mathcal{F}$ que vendrá dado por su ecuación normal $\langle w, \phi(x_i)\rangle + b = 0$. Para ello resolvemos el problema de optimización (1.25) de la Sección 1.2 con la nueva base de datos, quedando un problema de la siguiente forma:

$$\begin{aligned} \min_{w,b,\xi_i} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.a.} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq 1 + \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{1.40}$$

Desgraciadamente, en la práctica, la función ϕ es desconocida. Esto implica que la versión primal del Problema (1.40) es imposible de resolver. Es por ello, que se recurre a la versión dual. Para ello, adaptaremos el problema dual (1.33) a nuestra base de datos,

quedando el siguiente problema de optimización:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell \langle \phi(x_i), \phi(x_\ell) \rangle \\ \text{s.a.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \quad i = 1, \ldots, n \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \end{aligned} \tag{1.41}$$

Asumiendo que $\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*)$ son las variables de decisión óptimas del Problema (1.41), podemos encontrar las variables de decisión óptimas del problema primal (1.40), w^*, utilizando la expresión (1.29). De esta forma, tendremos que:

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i \phi(x_i) \tag{1.42}$$

Asimismo, la regla de clasificación para un nuevo punto x^{test} quedaría como:

$$\begin{aligned} \widehat{Y}(x^{test}) = & sign(\langle w^*, \phi(x^{test}) \rangle + b^*) = \\ & sign\left(\left\langle \sum_{i=1}^{n} \alpha_i^* y_i \phi(x_i), \phi(x^{test}) \right\rangle + b^* \right) = \\ & sign\left(\sum_{i=1}^{n} \alpha_i^* y_i \left\langle \phi(x_i), \phi(x^{test}) \right\rangle + b^* \right) \end{aligned} \tag{1.43}$$

donde la última igualdad se tiene por la bilinealidad del producto escalar.

Nótese que tanto a la hora de resolver el problema de optimización dual como para evaluar la regla de clasificación, no es necesario conocer la función ϕ, sino el producto escalar entre los puntos $\phi(x_i)$ y $\phi(x_\ell)$, $\forall i, \ell = 1, \ldots, n$. Es por ello, que podemos reescribir ese producto escalar, usando la llamada función kernel que definimos a continuación.

Definición 1.5 (Función kernel)**.** *Dada la función $\phi : \mathbb{R}^p \to \mathcal{F}$ definimos la función kernel $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ como:*

$$K(x, x') = \langle \phi(x), \phi(x') \rangle, \forall x, x' \in \mathbb{R}^p$$

Por construcción una función kernel es simétrica y bilineal al serlo también el producto escalar.

Con esta definición, el problema dual (1.41) puede reescribirse como:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell K(x_i, x_\ell) \\ \text{s.a.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \quad i = 1, \ldots, n \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \end{aligned} \tag{1.44}$$

Por tanto, como ϕ ya no aparece en el problema, no necesitamos saber su expresión de forma explícita. Solo es necesario conocer el producto escalar asociado K. El Problema (1.44) es un problema cuadrático cóncavo y con restricciones lineales, que puede

resolverse usando las técnicas de programación convexa de [8]. De igual forma, la regla de clasificación (1.43) para un nuevo punto x^{test} se puede reescribir en función del kernel como:

$$\widehat{Y}(x^{test}) = sign\left(\sum_{i=1}^{n} \alpha_i^* y_i \left\langle \phi(x_i), \phi(x^{test}) \right\rangle + b^*\right) = sign\left(\sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x^{test}) + b^*\right)$$

Existen varias funciones que se utilizan como kernel. Los ejemplos más comunes son:

$$\text{kernel lineal,} \qquad K(x, x') = \langle x, x' \rangle \tag{1.45}$$

$$\text{kernel polinómico de grado } d, \qquad K(x, x') = (1 + \langle x, x' \rangle)^d \tag{1.46}$$

$$\text{kernel de base radial,} \qquad K(x, x') = \exp(-\gamma ||x - x'||) \tag{1.47}$$

$$\text{kernel de red neuronal,} \qquad K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2) \tag{1.48}$$

Se puede observar que en el caso en el que tratemos con datos linealmente separables, resolver el Problema (1.33) es equivalente a resolver el Problema (1.44) con el kernel lineal. Para el caso donde los datos no se pueden separar linealmente, lo más común es usar el kernel de base radial ajustando apropiadamente el parámetro γ. Otras opciones alternativas son el kernel polinómico y la red neuronal ajustando correctamente sus parámetros.

Capítulo 2

Selección de variables en SVM

2.1. Introducción a las estrategias de selección de variables

En la actualidad, es muy fácil disponer de una gran cantidad de datos que podemos utilizar para crear modelos matemáticos de predicción utilizando técnicas como la de *Support Vector Machine* descrita en el Capítulo 1. El hecho de tener un gran volumen de datos implica considerar un número elevado de variables explicativas f_j. Esto conlleva en ocasiones a la construcción de modelos muy complejos y difíciles de interpretar. Es por ello que parece razonable desarrollar estrategias que seleccionen las variables explicativas más importantes de acuerdo a cierto criterio determinado por el usuario.

Por tanto, el objetivo principal de la selección de variables será elegir un subconjunto, que puede ser de un tamaño específico d, de las variables originales $P = \{f_1, \ldots, f_p\}$, de manera adecuada para facilitar la recolección de datos, reducir el espacio de almacenamiento de los mismos y el tiempo de entrenamiento. Además, la selección de variables será útil para interpretar el problema y mejorar la precisión en la clasificación. Para ilustrar el concepto de selección de variables, usaremos el Ejemplo 0.1. Supongamos que, además de la variable explicativa *altura* también disponemos de información sobre el peso de los individuos encuestados. El objetivo, de nuevo, será determinar si una persona es alta o baja. En este sencillo ejemplo, es fácil ver que de las dos variables disponibles, altura y peso, sólo una es realmente útil para la predicción. Esta variable es la altura. En consecuencia, el objetivo será construir una metodología que seleccione *altura* como variable importante, en lugar de *peso*. De hecho, después de realizar el proceso de selección de variables solo usaríamos la variable altura (interpretación más clara del problema). Además, tendríamos que si queremos saber si una persona es alta o baja no necesitamos recoger el dato de su peso (facilitar la recolección de datos) y, finalmente, si disponemos de una muestra de gran tamaño no usaremos los datos de peso ni sus respectivos cálculos al generar el modelo (reducir tiempo de entrenamiento).

Existen tres estrategias principales para la selección de variables. Estas metodologías se conocen por sus nombres en inglés, como: *filter*, *wrapper* y *embedded* [24, 33]. En las siguientes secciones, se darán más detalles sobre las distintas estrategias.

2.2. Métodos *filter*

Los métodos *filter* se utilizan como paso previo a la aplicación de algoritmos de *machine learning* (como por ejemplo, la estrategia SVM del Capítulo 1). La metodología *filter* elimina información de los datos de acuerdo a sus propiedades estadísticas.

Están basados en la evaluación de una cierta métrica, calculada directamente a partir de los datos, sin que exista una interacción con los predictores que finalmente se usarán en la base de datos con un número reducido de variables. Desde el punto de vista computacional, estos métodos son los menos costosos, pues, en general, se reduce al cálculo de algunas métricas estadísticas. Mientras que respecto a su precisión para hacer predicciones, los métodos *filters* no garantizan obtener el subconjunto de variables óptimo para el algoritmo de clasificación usado, puesto que este no interviene en ningún momento de su contribución [24, 50].

Sea $\mathcal{P}(P)$ el conjunto de todos los subconjuntos de P entonces, un filtro de características es una función $J : \mathcal{P}(P) \to \mathbb{R}$ que para un subconjunto de variables $S \subset \mathcal{P}(P)$ del conjunto total de variables P devuelve un índice de relevancia $J(S)$ que indica la importancia de ese subconjunto para una tarea, en nuestro caso para la tarea de clasificación. Estos índices se denominan 'métricas de selección de características', aunque no tengan las propiedades requeridas para llamarse métrica [24].

Un índice de relevancia bajo en el subconjunto S indicará que las *features* incluidas en S no son importantes para la clasificación.

En particular, podemos considerar conjuntos S con una única variable f_j, $j = 1, \ldots, p$. Esto nos permitirá establecer una relación de orden entre los índices de relevancia de las distintas *features* de la siguiente forma:

$$J(f_{j_1}) \leq J(f_{j_2}) \ldots \leq J(f_{j_p})$$

Así, la *feature* f_{j_1} es la menos importante, mientras que el papel de la *feature* f_{j_p} es crítico para la clasificación. Es por ello, que si nos quisiéramos quedar con las d variables explicativas más relevantes, tomaríamos aquellas con mayor índice de relevancia.

El hecho de eliminar variables una a una puede ser asequible cuando el valor de p no es muy alto. Sin embargo, si p es grande, esta tarea es computacionalmente intratable. Es por ello, que suele recurrirse al estudio de la eliminación de grupos de variables en lugar de variables individuales. Este proceso puede acelerar la carga computacional, aunque también tiene algunos inconvenientes. En primer lugar, decidir qué subconjuntos elegir y cuántos elementos consta cada conjunto es un problema que crece de forma exponencial con la dimensión de p. En segundo lugar, si disponemos de un conjunto de datos en el que existe una fuerte relación entre las variables, o dicho de otra forma en los que las variables tienen una alta correlación, esta estrategia no es buena, puesto que puede provocar que muchas de las variables seleccionadas pueden ser redundantes entre sí. Es decir, a priori, es decisión del usuario elegir entre velocidad de cómputo y forma de seleccionar las variables.

A continuación, se detallan algunos de los métodos *filter* más importantes.

2.2.1. *Fisher Criterion Score*

Uno de los métodos *filter* más usado y sencillo es el llamado por su nombre en inglés *Fisher Criterion Score* o *Fisher Score* que denotaremos por F. Esta estrategia calcula la importancia de cada una de las variables independientemente de las otras, a través de la correlación de la variable f_j con la variable respuesta y, [33]. Su objetivo principal

altura f_1	peso f_2	variable respuesta
170	85	alta
147	75	baja
150	77	baja
154	81	baja
175	79	alta
152	80	baja
173	82	alta
177	84	alta

Tabla 2.1: Datos del Ejemplo 2.1. Para cada observación tenemos en la columna izquierda los valores de la variable explicativa altura en centímetros f_1, en la columna del centro los valores de la variable explicativa peso en kilogramos f_2 y en la columna derecha la clase a la que pertenece.

es encontrar un subconjunto de características o *features*, de forma que se maximice la distancia entre los datos con diferentes clases, a la vez que se minimiza la distancia entre puntos con la misma etiqueta, [22, 44]. Matemáticamente, el *Fisher Criterion Score* de la variable f_j se expresa de la siguiente forma:

$$F(f_j) = \left| \frac{\mu_j^+ - \mu_j^-}{(\sigma_j^+)^2 + (\sigma_j^-)^2} \right| \tag{2.1}$$

donde μ_j^+ (respect. μ_j^-) representa la media de la variable j-ésima para la clase positiva (respect. negativa) y σ_j^+ (respect. σ_j^-) es la desviación estándar de la clase positiva (respect. negativa). El criterio de Fisher definido en (2.1) se puede interpretar de forma similar al cociente entre las varianzas entre-clase e intra-clase [24], comúnmente usada en el Análisis Discriminante Lineal, (o más conocido en inglés, por *Linear Discriminant Analysis*), [16]. Más concretamente, el numerador de (2.1) haría referencia a la varianza entre-clase, es decir, a la separación de los datos de clases diferentes. Por otro lado, el denominador se puede entender como la varianza intra-clase. En otras palabras, el denominador indica la dispersión de los datos de la misma clase. Por lo tanto, estamos interesados en seleccionar *features* tales que el numerador de (2.1) sea alto, y el denominador sea bajo. Consecuentemente, aquellas variables explicativas que tienen asociado un valor alto de F, serán consideradas importantes para la clasificación.

Ilustremos este concepto con un ejemplo:

Ejemplo 2.1. *Partiremos del Ejemplo 0.1. Supondremos que, además de los datos de la altura en centímetros (variable f_1) y de su clase, disponemos de los datos del peso de cada persona medidos en kilogramos (variable f_2). Toda esta información puede verse en la Tabla 2.1. Además, en la Figura 2.1 están representados los distintos individuos de la Tabla 2.1. En particular, los círculos azules muestran aquellas personas clasificadas como bajas, mientras que los puntos rojos hacen referencia a las personas altas.*

Supongamos ahora que estamos interesados en seleccionar la variable explicativa más relevante para hacer nuevas clasificaciones. A simple vista, podríamos intuir que la altura es la variable más importante para la clasificación. Esto es debido a que si proyectamos los datos sobre el eje de abscisas (variable altura) es más fácil distinguir ambas clases, que si proyectamos la base de datos contra el eje de ordenadas (variable peso).

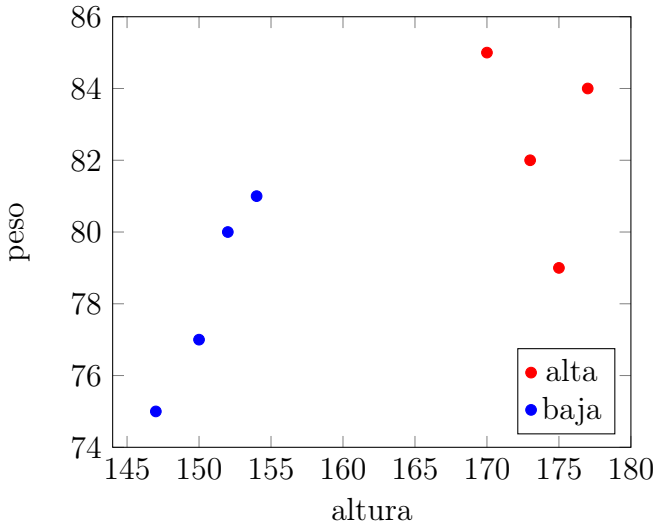


Figura 2.1: Dibujo ilustrativo del ejemplo, donde la variable altura tienen una varianza intra-clase baja y una varianza entre-clase alta. Por el contrario, la variable peso muestra una variable intra-clase es alta y la varianza entre-clase es baja.

Para comprobar si la intuición es cierta, será necesario calcular el Fisher Criterion Score *de ambas variables. Es decir,* $F(f_1)$ *y* $F(f_2)$*. Diremos que la variable más importante es aquella que tiene el* Fisher Score *más alto.*

Comenzaremos, calculando $F(f_1)$*. Para ello, de acuerdo a la Ecuación* (2.1) *será necesario calcular* μ_1^+, μ_1^-, σ_1^+ *y* σ_1^-*. En primer lugar, obtenemos que la media de la* feature *de las personas clasificadas como altas es* $\mu_1^+ = 173.75$*. A continuación, vemos que la media de la variable altura de los elementos clasificados como bajos es* $\mu_1^- = 150.75$*. Finalmente, vemos que la desviación típica de las personas altas y bajas es* $\sigma_1^+ = \sigma_1^- = 2.59$*. Con estos valores, y evaluando la Ecuación* (2.1)*, tenemos que* $F(f_1) = 4.44$*. Análogamente, calculamos el criterio de Fisher para la variable peso,* $F(f_2)$*. Para ello tenemos que* $\mu_2^+ = 82.5$, $\mu_2^- = 78.25$, $\sigma_2^+ = 2.29$ *y* $\sigma_2^- = 2.38$*. Por lo que* $F(f_2) = 0.91$*. Dado que* $F(f_1) > F(f_2)$*, concluimos que la variable explicativa, altura, es la más relevante para la clasificación, tal y como habíamos intuido.*

En esta sección, nos hemos centrado en el cálculo del *Fisher Criterion Score* para problemas de clasificación binaria y para *features* individuales. Sin embargo, este criterio se puede extender a problemas de clasificación con más de dos clases. También se puede calcular este método para conjuntos de variables no unitarios. Este enfoque queda fuera del alcance de este trabajo. Es por ello que para más información sobre este tema, se recomienda consultar [22].

2.2.2. Algoritmo *Relief*

El algoritmo *Relief* fue propuesto por Kira y Rendell en 1992, [31]. El éxito del algoritmo se debe al hecho de que es rápido, fácil de entender e implementar, [36]. El algoritmo *Relief* utiliza un enfoque basado en el algoritmo de k-vecinos más próximos, [28], presentado en la introducción de este trabajo.

El método de k-vecinos más próximos consiste en utilizar las observaciones del conjunto de entrenamiento $\{(x_1, y_1) \dots (x_n, y_n)\}$ que están más cerca del dato x^{test} para hacer predicción $\widehat{Y}(x^{test})$. Estas observaciones son precisamente los vecinos cercanos. De esta

forma, construimos la siguiente regla de clasificación:

$$\widehat{Y}(x^{test}) = sign\left(\frac{1}{k}\sum_{x_i \in N_k(x^{test})} y_i\right)$$

donde $N_k(x^{test})$ es el conjunto de los k puntos más cercanos a x^{test} en el conjunto de entrenamiento según la norma euclídea y $sign(a)$ es 1 si $a \geq 0$ y -1 si $a < 0$.

La idea del algoritmo *Relief* es sencilla. Dado un punto x, se selecciona el dato más cercano a su misma clase denotado por $H(x)$, así como la observación más cercana con diferente etiqueta, que denotaremos por $M(x)$. Nótese que la notación $H(x)$ y $M(x)$ provienen del inglés *nearest-hit* y *nearest-miss*, respectivamente. La cercanía de un punto x a $H(x)$ y $M(x)$ va a depender de la distancia elegida. En este trabajo, tomaremos la distancia Manhattan, o ℓ_1 dada por la suma de los valores absolutos de las diferencias entre las variables explicativas. En otras palabras, dados dos puntos x_i y x_ℓ, diremos que la distancia Manhattan viene dada por:

$$d(x_i, x_\ell) = \sum_{j=1}^{p} |x_{ij} - x_{\ell j}| \tag{2.2}$$

A pesar de haber hecho esta elección, podríamos haber tomado otras distancias alternativas como la euclídea. De hecho, en este caso, los resultados obtenidos serían muy similares, tal como puede verse en [32, 46]. Para cada variable explicativa f_j, $j = 1, \ldots, p$, el algoritmo *Relief* construye un índice de relevancia $J(f_j)$ de la siguiente forma: Si la diferencia de valores de la *feature* f_j entre x y $H(x)$ es muy grande, significa que esa variable explicativa separa dos observaciones de la misma clase, lo cual no es deseable. Es por ello, que el índice $J(f_j)$ disminuye su valor. Por el contrario, si los valores de x y $M(x)$ en la componente f_j son muy diferentes, entonces implicaría que esa *feature* está separando datos de clases distintas, lo cual es precisamente lo que estamos buscando.

Esta diferencia de valores en la variable explicativa f_j entre las observaciones x_i y x_ℓ será denotada por $diff(f_j, x_i, x_\ell)$. La expresión de $diff(f_j, x_i, x_\ell)$ dependerá de si la *feature* f_j es cuantitativa o cualitativa. En el primer caso, tendremos:

$$diff(f_j, x_i, x_\ell) = \frac{|x_{ij} - x_{\ell j}|}{\max\limits_{r=1,\ldots,n} x_{rj} - \min\limits_{r=1,\ldots,n} x_{rj}} \tag{2.3}$$

mientras que en el segundo será:

$$diff(f_j, x_i, x_\ell) = \begin{cases} 0, & \text{si} \quad x_{ij} = x_{\ell j} \\ 1, & \text{si} \quad x_{ij} \neq x_{\ell j} \end{cases}$$

Observar que $diff(f_j, x_i, x_\ell) \in [0, 1]$ tanto en el caso en que la variable f_j sea cuantitativa como cualitativa. En el caso de ser cualitativa es obvio porque la definición de $diff(f_j, x_i, x_\ell)$ solo tomas valores 0 y 1. Y en el caso de una variable cuantitativa, tanto el numerador como el denominador son mayores o iguales a 0, por lo que, la diferencia es mayor o igual a 0. Y el denominador $\max\limits_{r=1,\ldots,n} x_{rj} - \min\limits_{r=1,\ldots,n} x_{rj}$ toma el valor máximo que puede llegar a alcanzar el numerador $|x_{ij} - x_{\ell j}|$, en consecuencia, la diferencia es menor o igual a 1.

Por lo tanto, dado un vector x_{i_0}, y sus dos datos más cercanos de igual y distinta clase, $H(x_{i_0})$ y $M(x_{i_0})$, respectivamente, tendríamos que el índice de relevancia viene dado por

$$J(f_j) = diff(f_j, x_{i_0}, M(x_{i_0})) - diff(f_j, x_{i_0}, H(x_{i_0}))$$

Nótese que la expresión anterior depende de la elección del vector x_{i_0}. Por ello, con el objetivo de obtener un índice de relevancia más robusto, se suele seleccionar m instancias de entre las n disponibles, y tomar la media de los valores obtenidos en cada una de las m observaciones. De esta forma, tendríamos que el índice de relevancia del algoritmo *Relief* de la *feature* f_j viene dado por:

$$J(f_j) = \sum_{i=1}^{m} \frac{diff(f_j, x_i, M(x_i))}{m} - \sum_{i=1}^{m} \frac{diff(f_j, x_i, H(x_i))}{m} \tag{2.4}$$

Además, el índice de relevancia $J(f_j)$ se encuentra en $[-1, 1]$, debido a que $diff(f_j, x_i, M(x_i))$ y $diff(f_j, x_i, H(x_i))$ toma valores en $[0, 1]$, para $i = 1, \ldots, m$ y sus medias también pertenecen al intervalo $[0, 1]$.

En el artículo escrito por Kira y Rendell, [31], introducían un umbral de relevancia τ y admitían como variables relevantes f_j aquellas tal que $J(f_j)$ sea mayor que τ. Sin embargo, es suficiente con ordenar las variables explicativas de acuerdo a los valores de J. A continuación, se elegirá un subconjunto de variables de tamaño d, en función de las limitaciones de los algoritmos de clasificación que se aplicaran posteriormente, [36, 46].

La eficiencia del algoritmo se ha atribuido al hecho de no explorar explícitamente subconjuntos de características y no tratar de identificar un tamaño mínimo óptimo para el subconjunto de características, [46].

Ejemplo 2.2. *Aplicaremos el método* Relief *en los datos del Ejemplo 2.1 presentes en la Tabla 2.1. Tomaremos* $m = 2$ *y supongamos que aleatoriamente se seleccionan las instancias* $x_1 = (170, 85)$ *y* $x_6 = (152, 80)$. *Para* x_1, *una observación clasificada como alta, calculamos las diferencias de valores* $diff(f_j, x_1, x_\ell)$ *para* $j = 1, 2$ *y* $\ell = 2, \ldots, 8$ *así como la distancia Manhattan para obtener los puntos* $H(x_1)$ *y* $M(x_1)$. *Obtenemos de la Tabla 2.2 que* $H(x_1) = x_7$ *y* $M(x_1) = x_4$. *Del mismo modo, para* x_6, *que está clasificada como baja, obtenemos de la Tabla 2.3 que* $H(x_6) = x_4$. *Además, óbservese que* $M(x_6)$ *puede tomar tanto el valor* x_1 *como* x_7, *ya que ambos puntos están tienen la misma distancia a* x_6 *(ver última columna de la Tabla 2.3 para más detalles). En estos situaciones de empate, se elige aleatoriamente una de las dos opciones. En nuestro caso, tomaremos* $M(x_6) = x_7$. *En la Figura 2.2 podemos ver las observaciones y los vecinos más próximos de las instancias* x_1 *y* x_6. *Ahora calculamos los índices de relevancia según la Ecuación (2.4):*

$$J(f_1) = \frac{0.53 + 0.7}{2} - \frac{0.1 + 0.07}{2} = 0.7$$
$$J(f_2) = \frac{0.4 + 0.2}{2} - \frac{0.3 + 0.1}{2} = 0.1$$

Dando como resultado que según el método Relief *es preferible usar* f_1 *antes que* f_2, *puesto que* $J(f_1) > J(f_2)$. *Este hecho es intuitivo sabiendo que* f_1 *se corresponde con la variable altura y* f_2 *es el peso.*

x_i	y_i	$diff(f_1, x_1, x_i)$	$diff(f_2, x_1, x_i)$	$d(x_1, x_i)$
x_2	baja	0.77	1	33
x_3	baja	0.67	0.8	28
x_4	baja	0.53	0.4	20
x_5	alta	0.17	0.6	11
x_6	baja	0.6	0.5	23
x_7	alta	0.1	0.3	6
x_8	alta	0.24	0.1	8

Tabla 2.2: Tabla con diferencias de valores entre la instancia x_1, clasificada como alta, y el resto de observaciones. La última columna es la distancia Manhattan que corresponde a la Ecuación (2.2) y vemos que $H(x_1) = x_7$ y $M(x_1) = x_4$.

x_i	y_i	$diff(f_1, x_6, x_i)$	$diff(f_2, x_6, x_i)$	$d(x_6, x_i)$
x_1	alta	0.6	0.5	23
x_2	baja	0.17	0.5	10
x_3	baja	0.07	0.3	5
x_4	baja	0.07	0.1	3
x_5	alta	0.77	0.1	24
x_7	alta	0.7	0.2	23
x_8	alta	0.83	0.4	29

Tabla 2.3: Tabla con diferencias de valores entre la instancia x_6, clasificada como baja, y el resto de observaciones. La última columna es la distancia Manhattan que corresponde a la Ecuación (2.2) y vemos que $H(x_6) = x_4$ y $M(x_6) = x_7$

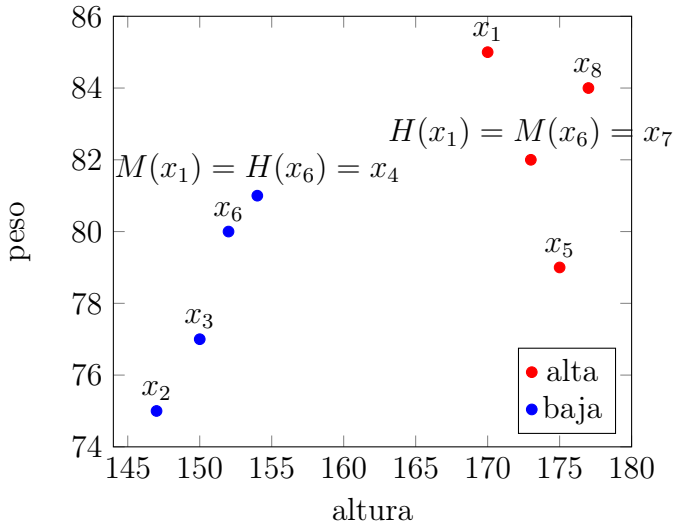


Figura 2.2: Dibujo ilustrativo del Ejemplo 2.2, donde vemos los vecinos más próximos de los ejemplos x_1 y x_6. Recordemos que la distancia empleada en este algoritmo no es la euclídea, sino la distancia Manhattan, por lo que x_8 no es el dato más cercano a x_1 aunque pueda parecerlo a simple vista.

2.2.3. Correlation based Feature Selection (CFS)

La selección de características basada en la correlación (conocida en inglés como *Correlation based Feature Selection*, CFS) es un algoritmo tipo *filter* que identifica los subconjuntos de características importantes de acuerdo a una medida de correlación, en lugar de evaluar las *features* de forma individual como se había hecho hasta ahora en las Secciones 2.2.1 y 2.2.2.

En general, la relevancia de un grupo de *features* en la tarea de clasificación aumenta con la correlación entre características y la clase, y, y disminuye con la correlación entre las *features* [30, 50]. Para el método CFS, dado un subconjunto de variables S de tamaño κ ($\kappa < p$), denotamos por $\overline{\rho_{yf}}$ a la media de las correlaciones entre cada característica en S y el atributo de clase y $\overline{\rho_{ff}}$ a la correlación media entre cada uno de los posibles $\binom{\kappa}{2}$ pares de características en S. Con esta notación, construimos nuestra función de evaluación de la siguiente manera:

$$J(S) = \frac{\kappa\overline{\rho_{yf}}}{\sqrt{\kappa + (\kappa - 1)\overline{\rho_{ff}}}} \tag{2.5}$$

donde numerador se puede interpretar como un indicador de cuán predictivo es el conjunto de características y el denominador se puede interpretar como un indicador de cuán redundantes son las características en S, [26, 41]. La función de evaluación (2.5) es, de hecho, el Coeficiente de correlación de Pearson, donde se han estandarizado todas las variables [24, 26, 27].

El problema que queda es desarrollar formas adecuadas de medir la correlación entre la clase y y la variable explicativa f_j y la intercorrelación entre las distintas *features*. El algoritmo CFS emplea una correlación, basada en el concepto de entropía, conocida como incertidumbre simétrica SU (del inglés *symmetrical uncertainty*) cuando se trata de problemas de clasificación y usa el coeficiente lineal de Pearson cuando se trata de problemas de regresión, [27]. Dado que este trabajo se centra en los métodos de selección de variables asociadas al Support Vector Machine (que es una forma de resolver problemas de clasificación), a continuación solo enunciaremos la definición de incertidumbre simétrica, SU.

Definición 2.1. *Dadas dos variables aleatorias X y Z discretas, denotamos a su incertidumbre simétrica por $SU(X, Z)$ y la definimos como:*

$$SU(X, Z) = 2\left[\frac{IG(X|Z)}{E(X) + E(Z)}\right]$$

donde $E(X)$ es la entropía de la variable X e $IG(X|Z)$ es la ganancia de información (en inglés information gain, IG) de la variable X respecto de la variable Z.

Obsérvese que en la Definición 2.1, además de ser necesarios los conceptos de entropía y ganancia de información, que explicaremos más adelante, se trabaja bajo la hipótesis de que las dos variables aleatorias sean discretas. Sin embargo, las tareas de aprendizaje supervisado, y en particular la clasificación, a menudo implican diferentes tipos de variables explicativas, cualquiera de las cuales puede ser cuantitativa o cualitativa, o lo que es lo mismo, continua o discreta. Es por ello que para poder aplicar la incertidumbre simétrica SU, en problemas de clasificación, necesitamos aplicar un preprocesamiento a la base de datos para discretizar las variables cuantitativas. Se suele aplicar un algoritmo de discretización propuesto por Fayyad e Irani (1993), [18] como paso previo al método [27, 41, 52].

Una vez hecha esta aclaración, pasaremos a definir la entropía de una variable:

	$Z=0$	$Z=1$	$Z=2$
$X=0$	1/6	1/15	4/15
$X=1$	1/6	4/15	1/15

Tabla 2.4: Tabla con los valores de la función de probabilidad de la variable conjunta (X, Z) del Ejemplo 2.3.

Definición 2.2 (Entropía)**.** *Dada una variable aleatoria discreta X que toma n valores y su función de probabilidad P, se define su entropía $E(X)$ como:*

$$E(X) := -\sum_{i=1}^{n} P(x_i)\log_2(P(x_i))$$

En consecuencia, la entropía conjunta de dos variables aleatorias discretas X y Z que toman n y m valores, respectivamente, se define como:

$$E(X,Z) := -\sum_{i=1}^{n}\sum_{\ell=1}^{m} P(x_i, z_\ell)\log_2(P(x_i, z_\ell))$$

Y la entropía de X dado unos valores observados de la variable Z es:

$$E(X|Z) := E(X,Z) - E(Z)$$

Veamos un ejemplo de entropía:

Ejemplo 2.3. *Sea X una variable aleatoria que sigue una distribución uniformemente discreta en un conjunto D_X formado por n puntos. Es decir, $P(x) = \frac{1}{n}$ si $x \in D_X$ y cero en caso contrario. Por tanto, la entropía de X viene dado por:*

$$\begin{aligned} E(X) &= -\sum_{i=1}^{n} P(x_i)\log_2(P(x_i)) = -\sum_{i=1}^{n}\frac{1}{n}\log_2\left(\frac{1}{n}\right) = \\ &= -n\frac{1}{n}\log_2\left(\frac{1}{n}\right) = -\log_2\left(\frac{1}{n}\right) = \log_2(n) \end{aligned} \tag{2.6}$$

Supongamos que $D_X = \{0, 1\}$ y además disponemos de una variable aleatoria Z que se distribuye uniformemente en el conjunto $D_Z = \{0, 1, 2\}$ y la función de probabilidad de la distribución conjunta (X, Z) vienen dada por la Tabla 2.4. Entonces, la entropía $E(X, Z)$ viene dada por:

$$E(X,Z) = -\left(\frac{2}{6}\log_2\left(\frac{1}{6}\right) + \frac{2}{15}\log_2\left(\frac{1}{15}\right) + \frac{8}{15}\log_2\left(\frac{4}{15}\right)\right) = 2.39958$$

Por otro lado, como Z es uniformemente discreta en 3 puntos ya sabemos por la Ecuación (2.6) que $E(Z) = \log_2(3)$. Por tanto, la entropía de la variable X condicionada a Z es:

$$E(X|Z) := E(X,Z) - E(Z) = 2.39958 - \log_2(3) = 0.81461$$

Si no se conoce la distribución conjunta de dos variables X y Z, pero se conoce la condicionada, puede usarse la siguiente proposición para calcular la entropía condicionada:

Proposición 2.1. *Sean X y Z dos variables aleatorias discretas que toman n y m valores, respectivamente. La entropía de X dado unos valores observados de la variable Z es equivalente a:*

$$E(X|Z) = -\sum_{\ell=1}^{m} P(z_\ell) \sum_{i=1}^{n} P(x_i|z_\ell) \log_2(P(x_i|z_\ell))$$

donde $P(z_\ell)$ es la probabilidad a priori y $P(x_i|z_\ell)$ la probabilidad a posteriori de X dado un valor de Z.

Demostración. Aplicando las definiciones de $E(X|Z)$, $E(X,Z)$ y $E(Z)$ vistas en la Definición 2.2 y reescribiéndolas en un sumatorio común tenemos:

$$E(X|Z) = E(X,Z) - E(Z) = -\sum_{\ell=1}^{m}\sum_{i=1}^{n} P(x_i, z_\ell) \log_2(P(x_i, z_\ell)) + \sum_{\ell=1}^{m} P(z_\ell) \log_2(P(z_\ell)) = \sum_{\ell=1}^{m} \left[P(z_\ell) \log_2(P(z_\ell)) - \sum_{i=1}^{n} P(x_i, z_\ell) \log_2(P(x_i, z_\ell)) \right]$$

Aplicando la regla del producto o probabilidad compuesta, es decir, $P(x_i, z_\ell) = P(z_\ell)P(x_i|z_\ell)$ dentro del logaritmo y por la propiedad de la suma de los logaritmos, $\log(ab) = \log(a) + \log(b)$ para a y b positivos, tenemos que:

$$E(X|Z) = \sum_{\ell=1}^{m} \left[P(z_\ell) \log_2(P(z_\ell)) - \sum_{i=1}^{n} (P(x_i, z_\ell) \log_2(P(z_\ell)) + P(x_i, z_\ell) \log_2(P(x_i|z_\ell))) \right] = \sum_{\ell=1}^{m} \left[P(z_\ell) \log_2(P(z_\ell)) - \log_2(P(z_\ell)) \sum_{i=1}^{n} P(x_i, z_\ell) - \sum_{i=1}^{n} P(x_i, z_\ell) \log_2(P(x_i|z_\ell)) \right]$$

donde la última igualdad se da tras descomponer el sumatorio interior en dos sumas. Como consecuencia de la regla del producto y el teorema de la probabilidad total tenemos que, para un z_ℓ fijo, $\sum_{i=1}^{n} P(x_i, z_\ell) = \sum_{i=1}^{n} P(z_\ell|x_i)P(x_i) = P(z_\ell)$, por lo que, el término $P(z_\ell) \log_2(P(z_\ell))$ se cancela con $\log_2(P(z_\ell)) \sum_{i=1}^{n} P(x_i, z_\ell)$ y aplicando nuevamente $P(x_i, z_\ell) = P(z_\ell)P(x_i|z_\ell)$ obtenemos el resultado buscado:

$$E(X|Z) = \sum_{\ell=1}^{m} \left[-\sum_{i=1}^{n} P(x_i, z_\ell) \log_2(P(x_i|z_\ell)) \right] = -\sum_{\ell=1}^{m} P(z_\ell) \sum_{i=1}^{n} P(x_i|z_\ell) \log_2(P(x_i|z_\ell))$$

□

Esta proposición la usaremos más adelante para conocer el valor de la incertidumbre simétrica de una variable aleatoria discreta X consigo misma, es decir, $SU(X,X)$.

Además de la entropía, para formalizar correctamente la Definición 2.1, necesitamos conocer el concepto de ganancia de información de una variable X respecto a otra variable Z:

Definición 2.3. *La ganancia de información de una variable aleatoria discreta X dada por otra variable discreta Z se denota por $IG(X|Z)$ y viene dada por:*

$$IG(X|Z) := E(X) - E(X|Z)$$

donde $E(X)$ y $E(X|Z)$ denotan respectivamente la entropía y la entropía condicionada.

Proposición 2.2. *La ganancia de información IG es simétrica.*

Demostración. Para probar que $IG(X|Z) = IG(Z|X)$, tenemos que demostrar que $E(X) - E(X|Z) = E(Z) - E(Z|X)$. Esto se puede deducir fácilmente de la definición de entropía condicionada, ya que $E(X) - E(Z|X) = E(X) - (E(X,Z) - E(Z)) = E(Z) - (E(X,Z) - E(X)) = E(Z) - E(Z|X)$. □

Ejemplo 2.4. *Continuando con los datos del Ejemplo 2.3 calculamos $IG(X|Z)$:*

$$IG(X|Z) = E(X) - E(X|Z) = \log_2(2) - 0.81461 = 1 - 0.81461 = 0.18539$$

Y la incertidumbre simétrica entre X y Z es:

$$SU(X,Z) = 2\left[\frac{IG(X|Z)}{E(X)+E(Z)}\right] = 2\left[\frac{0.18539}{\log_2(2)+\log_2(3)}\right] = 0.14344$$

Observar que siempre se cumple que $SU(X,X) = 1$, ya que, sabiendo que $p(x_i|x_\ell) = 1$ si $i = \ell$ y 0 en otro caso, por la Proposición 2.1 tenemos:

$$E(X|X) = -\sum_{\ell=1}^{n} P(x_\ell) \sum_{i=1}^{n} P(x_i|x_\ell)\log_2(P(x_i|x_\ell)) = -\sum_{i=1}^{n} P(x_i)\log_2(1) = 0$$

por lo que $IG(X|X) = E(X) - E(X|X) = E(X)$ y en consecuencia

$$SU(X,X) = 2\left[\frac{IG(X|X)}{E(X)+E(X)}\right] = 2\left[\frac{E(X)}{2E(X)}\right] = 1$$

Observar, además, que por consecuencia de la Proposición 2.2 la incertidumbre simétrica, SU, es simétrica.

Veamos, a continuación, un ejemplo práctico del algoritmo CFS para un conjunto de 3 variables:

Ejemplo 2.5. *Sea P un conjunto de 3 características $P = \{f_1, f_2, f_3\}$. Sea y la variable respuesta. Es decir, la variable que indica la clase de cada dato. Supongamos, que hemos calculado la incertidumbre simétrica de todas las parejas y las tenemos guardadas en la Tabla 2.5. Calculamos el índice de relevancia del conjunto P por la Fórmula (2.5), para lo cual necesitamos la media de las correlaciones tanto para pares* feature-*class $\overline{\rho_{yf}}$ como para pares* feature-feature *$\overline{\rho_{ff}}$ del conjunto P:*

- *Para calcular $\overline{\rho_{yf}}$ tomamos las correlaciones de todas las parejas* feature-*class de la Tabla 2.5, $SU(f_1,y) = 0.95$, $SU(f_2,y) = 0.8$ y $SU(f_3,y) = 0.7$ y calculamos su media:*
$$\overline{\rho_{yf}} = \frac{0.95 + 0.8 + 0.7}{3} = 0.82$$
- *Para calcular $\overline{\rho_{ff}}$ calculamos la media de los valores $SU(f_1,f_2) = 0.5$, $SU(f_1,f_3) = 0.33$ y $SU(f_2,f_3) = 0.75$ de la Tabla 2.5:*
$$\overline{\rho_{ff}} = \frac{0.5 + 0.33 + 0.75}{3} = 0.53$$
- *Finalmente, empleamos la Fórmula (2.5):*
$$J(P) = \frac{\kappa\overline{\rho_{yf}}}{\sqrt{\kappa + (\kappa-1)\overline{\rho_{ff}}}} = \frac{3 * 0.82}{\sqrt{3 + 2 * 0.53}} = 1.22$$

	f_1	f_2	f_3	y
f_1	1	0.5	0.33	0.95
f_2	0.5	1	0.75	0.8
f_3	0.33	0.75	1	0.7
y	0.95	0.8	0.7	1

Tabla 2.5: Tabla con los valores de la incertidumbre simétrica de los diferentes pares de características del Ejemplo 2.5.

Para el conjunto $S = \{f_1, f_2\}$*, tenemos que*

$$\begin{aligned}\overline{\rho_{yf}} =& \frac{SU(f_1, y) + SU(f_2, y)}{2} = \frac{0.95 + 0.8}{2} = 0.875\\ \overline{\rho_{ff}} =& SU(f_1, f_2) = 0.5\\ J(S) =& \frac{\kappa\overline{\rho_{yf}}}{\sqrt{\kappa + (\kappa - 1)\overline{\rho_{ff}}}} = \frac{2 * 0.875}{\sqrt{2 + 1 * 0.5}} = 1.11\end{aligned}$$

Para el conjunto $S = \{f_1, f_3\}$*, obtenemos que*

$$\begin{aligned}\overline{\rho_{yf}} =& \frac{SU(f_1, y) + SU(f_3, y)}{2} = \frac{0.95 + 0.7}{2} = 0.825\\ \overline{\rho_{ff}} =& SU(f_1, f_3) = 0.33\\ J(S) =& \frac{\kappa\overline{\rho_{yf}}}{\sqrt{\kappa + (\kappa - 1)\overline{\rho_{ff}}}} = \frac{2 * 0.825}{\sqrt{2 + 1 * 0.33}} = 1.08\end{aligned}$$

Para el conjunto $S = \{f_2, f_3\}$*, obtenemos que*

$$\begin{aligned}\overline{\rho_{yf}} =& \frac{SU(f_2, y) + SU(f_3, y)}{2} = \frac{0.8 + 0.7}{2} = 0.75\\ \overline{\rho_{ff}} =& SU(f_2, f_3) = 0.75\\ J(S) =& \frac{\kappa\overline{\rho_{yf}}}{\sqrt{\kappa + (\kappa - 1)\overline{\rho_{ff}}}} = \frac{2 * 0.75}{\sqrt{2 + 1 * 0.75}} = 0.9\end{aligned}$$

Finalmente, para conjuntos unitarios, donde $\kappa = 1$*, tenemos que* $J(f_j) = SU(f_j, y)$ *por lo que* $J(f_1) = 0.95$*,* $J(f_2) = 0.8$ *y* $J(f_3) = 0.7$*. Concluyendo que el conjunto más relevante de entre todos los posibles subconjuntos de variables es* $p = \{f_1, f_2, f_3\}$*, pues es el que tiene mayor valor,* J*. Sin embargo, con este conjunto no conseguiremos eliminar variables, pues coincide con el conjunto original. Si nos centramos en los subconjuntos de tamaño 2, debemos quedarnos con las variables* $\{f_1, f_2\}$*, puesto que* $J(\{f_1, f_2\}) > J(\{f_1, f_3\}) > J(\{f_2, f_3\})$*. Para terminar, si quisiéramos seleccionar subconjuntos unitarios de variables, tendríamos que la* feature *más relevante es* f_1*, y la menos* f_3*, ya que* $J(f_1) > J(f_2) > J(f_3)$*.*

2.3. Métodos *wrapper*

A diferencia de lo que ocurre con los métodos *filter*, los métodos *wrapper* interaccionan con el algoritmo de clasificación que se esté usando para encontrar el subconjunto de variables más importante de acuerdo a su poder predictivo. Es decir, que distintos

algoritmos de clasificación con un mismo método *wrapper*, podrían dar lugar a distintos conjuntos de variables seleccionadas. Por ejemplo, volviendo a las estrategias de clasificación mencionadas en la introducción, tendremos que la misma metodología *wrapper* podría seleccionar distintas variables en función de si tratamos con $k$nn, árbol de decisión o SVM.

Desde el punto de vista computacional, las estrategias *wrapper* son más lentas que las *filter*, puesto que estas suelen hacer una búsqueda exhaustiva de todas las posibles combinaciones de variables.

Sin embargo, los métodos *wrapper* devuelven mejores resultados en términos de precisión que los *filter*, ya que en su ejecución tienen en cuenta el algoritmo de clasificación.

Si la función de evaluación $J(\cdot)$ usada para saber qué subconjunto de variables es bueno emplea la arquitectura del modelo predictor para la cual se seleccionan las variables, entonces, tenemos un enfoque *wrapper*. La interpretación de los valores de retorno de $J(\cdot)$ es tal que cuanto menor sea el valor, mejor será el subconjunto.

La función $J(\cdot)$ más utilizada es la validación cruzada con k hojas o *k-fold cross-validation*. Esta consiste en dividir el conjunto de todos los datos Ω en k subconjuntos de datos Ω_q de forma que $\cup_{q=1}^{k}\Omega_q = \Omega$ y $\Omega_q \cap \Omega_s = \emptyset$ si $q \neq s$. A continuación, se toma la unión de $k-1$ subconjuntos como el conjunto de entrenamiento (*training set*) y el último como conjunto de prueba (*test set*) cualquiera. Posteriormente, se anotarán los errores cometidos, es decir el número de elementos en *test set* donde la predicción de su etiqueta no coincide con el valor real. Se repetirá el proceso hasta que los k conjuntos se hayan empleado como conjunto de prueba. Finalmente, la media de los errores cometidos en los k conjuntos será la precisión del modelo según la validación cruzada (de k hojas), [6, 24]. Como cada ejemplo solo aparece una vez en el conjunto de pruebas, debido a que los Ω_q son disjuntos, el error de la validación cruzada ϵ_{CV} se escribe como:

$$\epsilon_{CV} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{Y}_{-i}(x_i))$$

donde $L(\cdot)$ es una función de pérdida que mide los errores cometidos e $\widehat{Y}_{-i}$ es el valor de la predicción en el modelo construido cuando se utilizó el conjunto de prueba que contenía la observación i. Nótese que en este trabajo, el modelo de predicción que se usará será el *Support Vector Machine*, explicado en el Capítulo 1. No obstante, en el caso general, se utilizará el modelo de clasificación que elija el usuario.

Una variante de la validación cruzada con k hojas es el procedimiento conocido como dejar-uno-fuera (del inglés *leave-one-out* (*LOO*)). Este procedimiento es equivalente a tomar $k = n$ en la validación cruzada. Es decir, se usa una sola observación como conjunto de prueba y se repite este proceso hasta que todas las observaciones han sido utilizadas como conjunto de prueba (de tamaño uno), [6, 24]. El error de la validación cruzada ϵ_{LOO} se escribe, de igual forma que ϵ_{CV}, como:

$$\epsilon_{LOO} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{Y}_{-i}(x_i))$$

donde $L(\cdot)$ una función de pérdida que mide los errores cometidos e $\widehat{Y}_{-i}$ es el valor predicho en el modelo construido cuando se utilizó solo la observación i como el conjunto de prueba.

Veamos un ejemplo de *leave-one-out*:

Ejemplo 2.6. *Sea Ω la base de datos dada por la Tabla 2.1 y sea $P = \{f_1, f_2\}$ con f_1 la variable altura, f_2 la variable peso y sea $k = n = 8$. Como estamos ante un problema de clasificación binaria tomamos como función de perdida $L(y_i, \widehat{Y}_{-i}(x_i)) = 1$ si $y_i \neq \widehat{Y}_{-i}(x_i)$ y $L(y_i, \widehat{Y}_{-i}(x_i)) = 0$ en caso contrario. Empezamos viendo el desempeño de un modelo constituido solo por la variable altura f_1. Primero eliminamos la variable f_2 y dividimos Ω en subconjuntos de tamaño 1, $\Omega_1 = \{(170, alta)\}, \Omega_2 = \{(147, baja)\}, \Omega_3 = \{(150, baja)\}, \Omega_4 = \{(154, baja)\}, \Omega_5 = \{(175, alta)\}, \Omega_6 = \{(152, baja)\}, \Omega_7 = \{(173, alta)\}, \Omega_8 = \{(177, alta)\}$. Luego, para cada $i = 1, \ldots, 8$ generamos un modelo SVM de kernel lineal usando como conjunto de entrenamiento $\cup_{q=1,q\neq i}^{8} \Omega_q$ y evaluamos la clase de x_i predicha por este modelo $\widehat{Y}_{-i}(x_i)$. En este caso, en el que solo empleamos la variable f_1, todos los modelos han predicho de forma correcta la clase del ejemplo, por lo que:*

$$\epsilon_{LOO} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{Y}_{-i}(x_i)) = \frac{1}{8}\sum_{i=1}^{8} 0 = 0$$

Sin embargo, obtenemos otro resultado distinto al utilizar solamente la variable f_2. Empezamos eliminando la variable f_1 de la base de datos y dividiéndolo en conjuntos unitarios $\Omega_1 = \{(85, alta)\}, \Omega_2 = \{(75, baja)\}, \Omega_3 = \{(77, baja)\}, \Omega_4 = \{(81, baja)\}, \Omega_5 = \{(79, alta)\}, \Omega_6 = \{(80, baja)\}, \Omega_7 = \{(82, alta)\}, \Omega_8 = \{(84, alta)\}$. Al realizar el proceso obtenemos casos donde se realiza mal la predicción:

1. *Al entrenar el modelo con el conjunto*

   $$\{(85, alta), (75, baja), (77, baja), (79, alta), (80, baja), (82, alta), (84, alta)\}$$

   y utilizar $\{(81, baja)\}$ como conjunto de prueba, se predice que $\{(81, baja)\}$ es alta.

2. *Entrenando el modelo con el conjunto de entrenamiento*

   $$\{(85, alta), (75, baja), (77, baja), (81, baja), (80, baja), (82, alta), (84, alta)\}$$

   se predice que $\{(79, alta)\}$ es baja.

3. *Cuando se entrena el modelo con el conjunto*

   $$\{(85, alta), (75, baja), (77, baja), (81, baja), (79, baja), (82, alta), (84, alta)\}$$

   la observación $\{(80, baja)\}$ se predice como alta.

Por lo tanto, el error cometido en el caso de modelos construidos con la variable peso f_2 es:

$$\epsilon_{LOO} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{Y}_{-i}(x_i)) = \frac{1+1+1}{8} = \frac{3}{8}$$

En conclusión, para desarrollar un modelo SVM de kernel lineal con una única variable elegiríamos la variable f_1 antes que f_2, ya que f_1 genera un error menor en la validación cruzada.

Los métodos *wrapper* se pueden dividir de acuerdo a su estrategia de búsqueda, es decir, en función del orden en el que se evalúan los subconjuntos de variables. Estas estrategias de búsquedas son: la búsqueda óptima, la búsqueda secuencial, y la búsqueda aleatoria.

2.3.1. Búsqueda óptima

En la estrategia de búsqueda óptima, nos centramos en encontrar el llamado subconjunto óptimo de variables. En otras palabras, queremos encontrar un subconjunto para el cual el procedimiento de estimación del error (por ejemplo, la validación cruzada) proporcione el menor valor posible en comparación con el resto de subconjuntos de variables. A continuación, daremos más detalles sobre dos estrategias de búsqueda óptima: la búsqueda exhaustiva, y la metodología de bifurcar y acotar.

Búsqueda exhaustiva

Una forma sencilla de encontrar el subconjunto óptimo es evaluar todos los subconjuntos posibles y quedarnos con aquel que tenga un menor error de predicción. Este enfoque se denomina búsqueda exhaustiva: dadas p variables explicativas, hay $2^p - 1$ subconjuntos por recorrer. Esta estrategia es viable desde el punto de vista computacional, cuando el valor de p es pequeño. Desafortunadamente, incluso para valores moderados de p es imposible evaluar todos los subconjuntos en un tiempo razonable. Por ejemplo, si $p = 10$, deberíamos de evaluar 1023 conjuntos, lo cual puede llegar a ser inviable.

Una forma de reducir la carga computacional puede ocurrir si se conoce de antemano que se deben elegir d variables entre las p posibles. Así, solo sería necesario evaluar los conjuntos de tamaño d. En consecuencia, solo habría que evaluar $\binom{p}{d}$ subconjuntos en lugar de $2^p - 1$. Por ejemplo, volviendo al caso anterior, si $p = 10$ y $d = 3$, tendríamos que comprobar la precisión en $\binom{10}{3} = 120$ subconjuntos, en lugar de los 1023 anteriores. Es decir, estaríamos reduciendo en aproximadamente un 90 % el número de subconjuntos a evaluar. Desafortunadamente, incluso en este caso, hay valores de p y d para los que el cálculo de $\binom{p}{d}$ es inviable.

Ejemplo 2.7. *Empleando como función de evaluación $J(\cdot)$ en el proceso de* leave-one-out*, tenemos que el Ejemplo 2.6 es una búsqueda exhaustiva donde $p = 2, d = 1$ y los únicos conjuntos de* features *de tamaño d son $\{f_1\}$ y $\{f_2\}$ con $J(\{f_1\}) = 0$ y $J(\{f_2\}) = 3/8$. Por tanto, el conjunto $\{f_1\}$ es el seleccionado por este método.*

Bifurcar y acotar

Si se desea un subconjunto de cierto tamaño d existe una estrategia óptima que potencialmente se ejecuta mucho más rápido que la búsqueda exhaustiva, desarrollada en [39] que solo se puede emplear cuando la función de evaluación $J(\cdot)$ es monótona. Entendemos por función monótona en este contexto que la adición de una variable nunca empeora un subconjunto [24], o dicho de otra forma que un subconjunto de características no debería ser mejor que cualquier conjunto más grande que contenga a dicho subconjunto. Formalmente, diremos que una función de evaluación J es monótona si dados dos subconjuntos de características S y S' tales que $S \subset S'$, entonces se tiene que $J(S) \geq J(S')$. Algunos ejemplos de funciones monótonas son medidas de distancia como la distancia Bhattacharyya y la divergencia [39].

La idea principal del algoritmo bifurcar y acotar se basa en evaluar un subconjunto S' formado por d variables y otro subconjunto S con un número arbitrario de variables $D > d$. Asumiremos que S' es más relevante que S. Es decir que $J(S) \geq J(S')$. Por lo tanto, debido a la monotonía de J, se tiene que cada subconjunto $\tilde{S}$ de S de tamaño d verifica que $J(\tilde{S}) \geq J(S)$. En consecuencia, se tiene que $J(\tilde{S}) \geq J(S) \geq J(S')$, $\forall \tilde{S} \subset S$. Por lo tanto, dados los conjuntos S y S' con las características anteriormente mencionadas,

no será necesario evaluar los subconjuntos de tamaño d de S, puesto que su función de evaluación siempre será peor que la de S, y consecuentemente de S'.

El algoritmo tiene una complejidad exponencial en el peor de los casos, lo que puede hacer que el enfoque no sea factible cuando hay disponible una gran cantidad de variables [24]. Este algoritmo no es muy útil en la práctica, porque la función de evaluación $J(\cdot)$ no suele ser monótona, como ocurre en el caso de la validación cruzada de k hojas, [24].

2.3.2. Selección secuencial

Debido a las dificultades con las estrategias óptimas, se han desarrollado estrategias de búsqueda que encuentren subconjuntos de variables razonablemente buenos sin tener que evaluar todos ellos.

Dentro de la búsqueda secuencial las dos estrategias más destacadas son: eliminación secuencial hacia atrás o selección secuencial hacia atrás (en inglés, *Sequential Backward Elimination*, SBE o *Sequential Backward Selection*, SBS, respectivamente) y selección secuencial hacia adelante (conocida por su nombre en inglés como *Sequential Forward Selection*, SFS).

El primer método, SBE, parte de un conjunto que incluye a todas y cada una de las variables de la base de datos y va eliminando, de una en una, las variables que dan mejor resultado en el ajuste del clasificador.

Por otro lado, el método SFS se inicializa con un conjunto de variables vacío. En cada iteración del algoritmo, se añade aquella variable que minimiza la métrica de evaluación $J(\cdot)$ de entre todas las variables que no pertenecen al subconjunto seleccionado.

A continuación, vamos a desarrollar ambas estrategias con más detalle.

Selección secuencial hacia atrás

La selección secuencial hacia atrás (SBS) o eliminación secuencial hacia atrás (SBE) parece haber sido el primer método de búsqueda sugerido para la selección de subconjuntos variables, [24].

SBS parte del conjunto original de *features*, $S = P$, es decir con todas las variables que inicialmente aparecen en la base de datos. A continuación, se inicia un proceso iterativo donde en cada paso se considera el conjunto $S \setminus \{f_j\}$, siendo f_j cada, una de las variables que pertenecen a S. Posteriormente, se evalúa esos conjuntos mediante la métrica de evaluación $J(\cdot)$, eliminando aquella *feature* f_j tal que el valor de $J(S \setminus \{f_j\})$ sea menor, [24]. Este proceso se repite hasta que se cumple un criterio de parada establecido, como puede ser el número de variables seleccionadas o la precisión del método de clasificación.

El Algoritmo 1 muestra un pseudocódigo de como funciona esta metodología.

Obsérvese que el método SBS es capaz de detectar cuando dos variables son más beneficiosas juntas, aunque no lo sean por separado. Esto es debido a que esta metodología evalúa cada variable en el contexto de las que permanecen en S, por lo que al evaluar $S \setminus \{f_{j'}\}$ para una variable relevante junto a otra de S, producirá un aumento en el valor de la función de evaluación $J(\cdot)$ respecto de una variable que sea relevante por sí misma, resultando, por lo tanto, que $f_{j'}$ no se elimine antes que las otras *features*.

Veremos con un ejemplo ilustrativo como funciona esta metodología.

Ejemplo 2.8. *Para ilustrar este método usaremos los datos de la Tabla 2.1. Además, a dichos datos le añadiremos una tercera variable de ruido. El objetivo de añadir dicha nueva* feature *es doble. Por un lado, realizaremos dos iteraciones del método SBS (en*

Algoritmo 1 *Sequential Backward Selection*, SBS

Función SBS(P, J, criterio de parada)
  $S := P$
  mientras no se verifique el criterio de parada **hacer**
    para todo $f_j \in S$ **hacer**
      Evaluar $J(S \setminus \{f_j\})$
    terminar para
    La variable eliminada es $j^* = \arg\min_j J(S \setminus \{f_j\})$
    $S := S \setminus \{f_{j^*}\}$
  terminar mientras
  devolver S
terminar Función

altura f_1	peso f_2	ruido f_3	variable respuesta
170	85	0.011	alta
147	75	0.34	baja
150	77	0.034	baja
154	81	0.092	baja
175	79	0.065	alta
152	80	0.358	baja
173	82	0.36	alta
177	84	0.757	alta

Tabla 2.6: Datos del Ejemplo 2.8. Para cada observación tenemos en la columna izquierda los valores de la variable explicativa altura en centímetros f_1, en la segunda columna los valores de la variable explicativa peso en kilogramos f_2, en la tercera columna una variable de ruido y en la última columna la clase a la que pertenece.

lugar de una) y, por lo tanto, mostraremos mejor cómo funciona este enfoque. Por otro lado, estamos 100 % seguros que la variable ruido debe ser eliminada con el método SBS, ya que es obvio que al ser ruido no nos proporciona información útil para predecir si una persona es alta o baja.

Para generar la variable de ruido han generado ocho observaciones siguiendo una probabilidad uniforme en el intervalo $(0, 1)$*. El conjunto de datos completo que incluye las variables altura y peso, así como la nueva variable de ruido y la variable respuesta pueden verse en la Tabla 2.6. Al ejecutar el método SBS hemos establecido que el criterio de parada sea tener un subconjunto de variables de tamaño 1.*

Partimos del conjunto inicial de variables $S = P = \{f_1, f_2, f_3\}$ *y empezamos evaluando todos los conjuntos de tamaño 2, es decir,* $\{f_1, f_2\}$*,* $\{f_2, f_3\}$ *y* $\{f_1, f_3\}$*. Obtenemos los siguientes resultados* $J(\{f_1, f_2\}) = 0$*,* $J(\{f_2, f_3\}) = 4/8 = 1/2$ *y* $J(\{f_1, f_3\}) = 0$*. Es decir, podemos eliminar tanto* f_2 *como* f_3 *en esta primera iteración. Eliminamos, por ejemplo,* f_2 *con lo que nuestro conjunto actual es* $S = \{f_1, f_3\}$*. Ahora evaluamos los conjuntos* $\{f_1\}$ *y* $\{f_3\}$*, obteniendo que* $J(\{f_1\}) = 0$ *y* $J(\{f_3\}) = 8/8 = 1$*. Por lo tanto, eliminamos* f_3 *de esta segunda iteración. Como hemos alcanzado el criterio de parada, concluimos que el subconjunto final de variables es* $S = \{f_1\}$*. Nótese que los resultados obtenidos son coherentes con la intuición que teníamos, puesto que la variable* f_1 *corresponde a la altura de los individuos, la cual es clave para determinar si una persona es alta o baja.*

Selección secuencial hacia adelante

El crecimiento secuencial o selección secuencial hacia adelante (SFS) es un método muy parecido a SBS. La principal diferencia es que SFS parte de un conjunto vacío denotado por S, en lugar de comenzar considerando todas las *features* como SBS. A continuación, inicializamos un proceso iterativo donde consideramos los conjuntos $S \cup \{f_j\}$ para cada variable f_j que no está en S. Evaluamos estos conjuntos mediante la función de evaluación $J(\cdot)$ y se añade a S aquella *feature* f_j que proporciona el menor valor de $J(S \cup \{f_j\})$. Repitiendo este proceso hasta que se cumple un criterio de parada establecido, como el número determinado de *features* seleccionadas o la precisión del método de clasificación. El pseudocódigo de cómo funciona esta metodología puede verse en el Algoritmo 2.

Algoritmo 2 *Sequential Forward Selection*, SFS

Función SFS(P, J, criterio de parada)
  $S := \emptyset$
  mientras no se verifique el criterio de parada **hacer**
    para todo $f_j \notin S$ **hacer**
      Evaluar $J(S \cup \{f_j\})$
    terminar para
    La variable añadida es $j^* = \arg\min_j J(S \cup \{f_j\})$
    $S := S \cup \{f_{j^*}\}$
  terminar mientras
  devolver S
terminar Función

Generalmente, cuando se establece un criterio de parada en un número específico de variables, el método SFS se ejecuta más rápido que el algoritmo SBS. Esto se debe al hecho de que, al comienzo de la búsqueda, cuando ambos algoritmos tienen muchos posibles conjuntos para evaluar en cada paso, SFS evalúa conjuntos de variables muy pequeños, lo que suele ser más rápido que evaluar los conjuntos casi completos que SBS tiene que hacer, [24].

Por ejemplo, supongamos que tenemos $p = 10$ variables en nuestra base de datos, y el criterio de parada tanto para SBS como para SFS es tener un subconjunto de 3 *features*.

En el primer método, SBS, comenzaremos evaluando el conjunto de 10 *features*, y luego eliminando progresivamente una a una aquellas variables con peor función de evaluación. Es decir, que tendríamos conjuntos de 9 variables, luego 8, y así sucesivamente hasta llegar al subconjunto deseado de cardinal 3.

Por otro lado, la metodología SFS comenzaría añadiendo al conjunto vacío inicial las variables una a una. De esta forma, comienza evaluando conjuntos de 1 variable, luego 2 y finalmente 3.

Además de que en el primer método SBS se deben hacer más iteraciones que en SFS (7 frente a 3), cabe esperar que las iteraciones de SBS, donde hay que considerar conjuntos casi completos de *features*, tengan una mayor carga computacional que SBS.

Por otro lado, SFS evalúa las variables en el contexto de solo aquellas variables que ya están incluidas en el conjunto, es decir, podría darse el caso de tener dos variables f_{j_1} y f_{j_2} que no sean muy relevantes por separado, pero sí cuando están juntas. Al evaluar $S \cup \{f_{j_1}\}$ o $S \cup \{f_{j_2}\}$ sin que la otra esté en el conjunto S no se producirá una mejora significativa en la función de evaluación $J(\cdot)$ con respecto a las variables que son relevantes

por sí solas. Por lo tanto, es posible que no pueda detectar una variable que es beneficiosa no por sí misma, sino solo con la información que contienen otras variables.

Ejemplo 2.9. *Dados los datos de la Tabla 2.6, vamos a aplicar el método SFS. Como criterio de parada, fijaremos a dos el número total de variables seleccionadas. Partimos del conjunto inicial de variables* $S = \emptyset$*, empezamos evaluando todos los conjuntos de tamaño 1, es decir,* $\{f_1\}, \{f_2\}$ *y* $\{f_3\}$*. Obtenemos los siguientes resultados* $J(\{f_1\}) = 0, J(\{f_2\}) = 3/8$ *y* $J(\{f_3\}) = 1$*, así que añadimos* f_1 *a nuestro conjunto de características seleccionado. Ahora evaluamos los conjuntos* $\{f_1, f_2\}$ *y* $\{f_1, f_3\}$*, obteniendo que* $J(\{f_1, f_2\}) = 0$ *y* $J(\{f_1, f_3\}) = 0$*. Por lo tanto, podemos añadir tanto* f_2 *como* f_3 *en esta segunda iteración, obteniendo dos conjuntos óptimos de tamaño 2 dependiendo de esta última elección. Observar que el hecho de que* $J(\{f_1\}) = J(\{f_1, f_2\}) = J(\{f_1, f_3\}) = 0$ *indica que las variables* f_2 *y* f_3 *no aportan información extra a la que se obtiene empleando* f_1*, es decir, solamente con* f_1 *se puede ajustar el modelo perfectamente. Por lo que de estar interesados en emplear el mínimo número de variables posibles, la opción más acertada sería tomar el conjunto* $\{f_1\}$*.*

2.3.3. Búsqueda aleatoria

Las estrategias de búsqueda antes mencionadas no tenían ninguna componente aleatoria, lo que las convierte en métodos deterministas que con la misma inicialización devuelven siempre los mismos resultados. Esto lleva a que el conjunto de variables finales sea sensible al conjunto de datos particular y puede cambiar de forma sustancial si falta, aunque sea un dato, o si estos se modifican ligeramente, [24].

Para ilustrar el hecho de que pequeñas modificaciones de los datos pueden alterar los resultados obtenidos con las metodologías deterministas, consideramos el siguiente ejemplo:

Ejemplo 2.10. *Partiremos de la base de datos de la Tabla 2.1, hemos visto en el Ejemplo 2.6 que* $J(\{f_1\}) = 0$ *y* $J(\{f_2\}) = 3/8$ *al realizar la validación cruzada con modelos de SVM de kernel lineal.*

Supongamos ahora que, debido a un error humano, se producen modificaciones en las observaciones x_4 *y* x_5*, de forma que la observación* x_4 *antes era* $(154, 81)$ *y ahora es* $(165, 78)$*. De igual manera, el elemento* x_5 *antes era* $(175, 79)$ *y ahora es* $(149, 85)$*. Nótese que las clases no se han modificado. En la Figura 2.3 puede verse la diferencia entre la base de datos anterior y la modificada.*

Obsérvese que con la modificación de los datos, la clase de los nuevos elementos viene determinada por la variable explicativa "peso". Esto es algo contraintuitivo, puesto que la etiqueta que queremos predecir es si una persona se considera alta o no. Es decir, que el error humano y el uso de metodologías deterministas ha provocado que lleguemos a conclusiones equivocadas. Este hecho además puede confirmarse con los siguientes resultados. Hemos entrenado la nueva base de datos usando un SVM lineal. Para hacer una comparativa justa con los resultados de la base anterior, se ha entrenado el modelo SVM dos veces. En cada ocasión se ha usado una feature *diferente para ajustar el modelo. A través de la precisión dada por el* leave-one-out*, hemos obtenido que* $J(\{f_1\}) = 2/8$ *y* $J(\{f_2\}) = 0$*. Es decir, contrario a lo que ocurría en el caso anterior (donde* $J(\{f_1\}) = 0$ *y* $J(\{f_2\}) = 3/8$*), tenemos que el peso es la variable más importante para determinar la clase.*

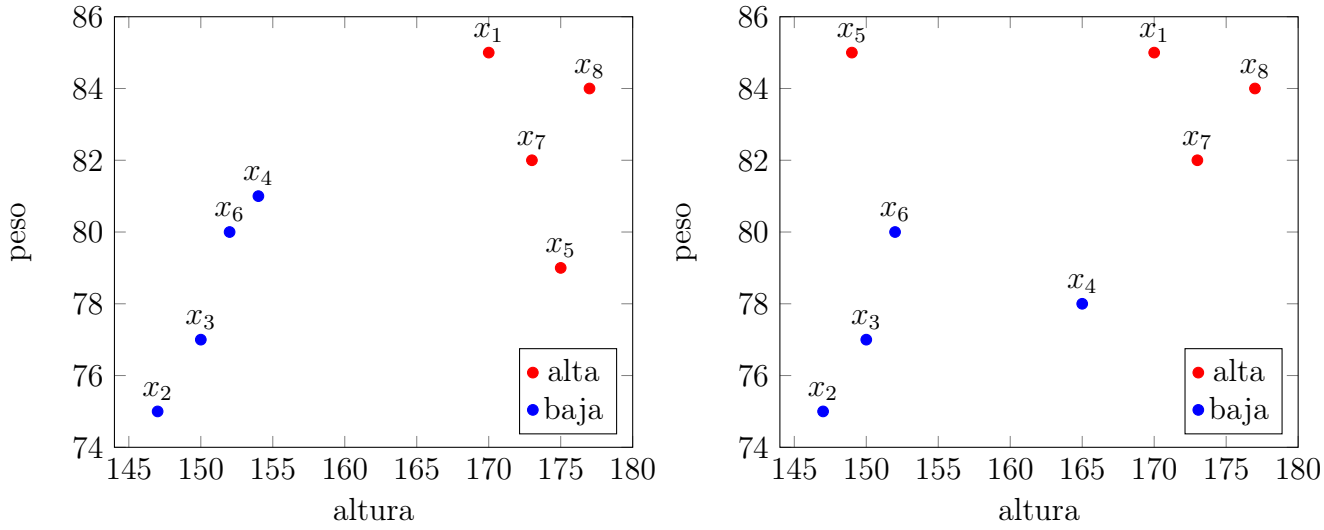


Figura 2.3: A la izquierda la base original de la Tabla 2.1. A la derecha la nueva base de datos modificada del Ejemplo 2.10 donde la variable peso es más relevante que la variable altura.

Dos metodologías usadas en la búsqueda aleatoria son el recocido simulado (*simulated annealing*) y los algoritmos genéticos.

Recocido simulado

La idea principal del recocido simulado (*Simulated annealing*, SA) se basa en el fenómeno físico por el cual la materia se enfría y se congela, hasta llegar a una estructura cristalina que minimiza la energía del cuerpo. El recocido se conoce como un proceso térmico para obtener estados de baja energía de un sólido en un baño térmico. El proceso consta de los siguientes dos pasos:

- Aumentar la temperatura del baño térmico hasta un valor máximo en el que el sólido se funda.
- Disminuir cuidadosamente la temperatura del baño térmico hasta que las partículas se acomoden en el llamado estado fundamental del sólido.

En la fase líquida, todas las partículas se disponen de forma aleatoria, mientras que en el estado fundamental del sólido, las partículas están dispuestas en una red muy estructurada, cuya energía correspondiente es mínima.

El algoritmo de recocido simulado o *Simulated annealing* imita la evolución de un sólido en un baño térmico hasta alcanzar el equilibrio térmico. Dado un estado actual s del sólido con energía E_s, se genera un estado posterior s' aplicando un mecanismo de perturbación que transforma el estado actual en otro estado mediante una pequeña distorsión, por ejemplo mediante el desplazamiento de una partícula. La energía del siguiente estado es $E_{s'}$. Si la diferencia de energía, $E_s - E_{s'}$, es menor o igual a 0, significa que el nuevo estado tiene menor energía que el anterior, que es justo el comportamiento que estamos buscando. Por lo tanto, se acepta el estado s' como estado actual. Si la diferencia de energía es mayor que 0 significa que el nuevo estado es "peor" que el anterior, ya que la energía es mayor. Sin embargo, en lugar de rechazar ese cambio, lo aceptaremos con cierta

probabilidad dada por

$$\exp\left(\frac{E_s - E_{s'}}{k_B T}\right) \tag{2.7}$$

donde T es la temperatura del baño y k_B a una constante física conocida como constante de Boltzmann, [1].

En nuestro caso particular de selección de variables, los estados de los sólidos s y s' representan los distintos conjuntos de variables y la energía de un estado E_s sería el valor de la función de evaluación $J(\cdot)$. Por lo que obtener un estado con energía mínima es equivalente a encontrar un conjunto de características óptimo.

El método empieza con un conjunto aleatorio de características S y una "temperatura" $T_0 > 0$ elevada, que servirá para determinar la probabilidad de aceptar cambios desfavorables en el conjunto S, de manera que un mismo cambio desfavorable será más fácil de aceptar a temperaturas altas que a temperaturas bajas. Continuamos añadiendo o eliminando aleatoriamente una variable en S obteniendo un nuevo conjunto S'. Si produce un mejor resultado con la función de evaluación $J(\cdot)$ aceptamos ese cambio. Por el contrario, si el cambio llega a un resultado peor usaremos una expresión análoga a (2.7), en función de la temperatura del paso actual T y la diferencia de valores de la evaluación con el conjunto anterior $J(S') - J(S)$, que denotaremos por ΔJ, como probabilidad para aceptar o rechazar el cambio, de modo que con altas temperaturas es más fácil aceptar el cambio que con bajas:

$$\exp\left(\frac{-\Delta J}{T}\right) \tag{2.8}$$

donde podemos dividir por T, ya que este valor es distinto de cero porque el algoritmo empieza con la temperatura $T_0 > 0$. Para descender la temperatura en los siguientes pasos la multiplicamos por una constante positiva menor que 1 previamente fijada, v. Obsérvese también que al estar en el caso de un cambio desfavorable, $\Delta J > 0$ lo que lleva a que el cociente $\frac{-\Delta J}{T} < 0$ y la Ecuación (2.8) toma valores entre 0 y 1. Debido a esto, en la práctica, para decidir si se acepta un cambio desfavorable se genera un número aleatorio r de manera uniforme en el intervalo $[0, 1]$. Se aceptará el cambio si y solo si r es menor que el valor de la Ecuación (2.8). Cuando se han ejecutado m iteraciones en la misma temperatura sin producirse mejoras en la evaluación del conjunto, procedemos a descender la temperatura como se ha indicado. Se repite el proceso hasta que la temperatura es más baja que una cierta cota T_1. Esta metodología no se queda atrapada en óptimos locales al empezar con temperatura alta. Aun así, es posible que al final de la búsqueda, encontremos un óptimo local, debido a que la baja temperatura ya no permite realizar cambios que produzcan peores valores de la función de evaluación $J(\cdot)$. Más detalles se pueden encontrar en el Algoritmo 3.

A continuación, ilustraremos este proceso con un ejemplo.

Ejemplo 2.11. *Sea $m = 2$, $T_0 = 10$, $T_1 = 3$ y $v = 0.4$. Dados los datos de la Tabla 2.6 vamos a aplicar el método SA. Comenzando con $T = 10$ y la solución aleatoria $S = \{f_2\}$ añadimos alguna característica que no esté en S de forma aleatoria, por ejemplo, f_3 obteniendo $S' = \{f_2, f_3\}$. Tenemos que $J(S) = J(\{f_2\}) = 3/8 < 4/8 = J(\{f_2, f_3\}) = J(S')$. Es decir, añadir la variable f_3 ha empeorado la función de evaluación. Esto tiene sentido, puesto que f_3 era la variable ruido que habíamos creado artificialmente. Por tanto, debemos generar un número aleatorio entre 0 y 1 y compararlo con la Ecuación (2.8) para*

Algoritmo 3 *Simulated annealing*, SA

Función SA(P, J, T_0, T_1, m, v)
Sea S un subconjunto aleatorio de P.
Iniciamos el algoritmo con la temperatura inicial $T := T_0$
mientras $T > T_1$ **hacer**
mientras no se realicen m iteraciones sin mejoras en la temperatura T **hacer**
$S' := S$
Aleatoriamente tomamos una variable f_j de P
$S' := S \cup \{f_j\}$ si $f_j \notin S$ o $S' := S \setminus \{f_j\}$ si $f_j \in S$.
Calculamos la diferencia de la métrica de evaluación $\Delta J := J(S') - J(S)$
si $\Delta J < 0$ **entonces**
$S := S'$
en otro caso
Generamos un número aleatorio $r \in [0, 1]$
si $r < \exp(-\Delta J/T)$ **entonces**
$S := S'$
terminar si
terminar si
terminar mientras
Cuando llevamos varias iteraciones en la misma temperatura sin mejoras bajamos la temperatura $T := T * v$
terminar mientras
devolver S
terminar Función

saber si aceptar o rechazar el cambio. El cálculo de la Ecuación (2.8) *sería en este caso:*

$$\exp\left(\frac{-\Delta J}{T}\right) = \exp\left(\frac{-\left(\frac{4}{8} - \frac{3}{8}\right)}{10}\right) = \exp\left(-\frac{1}{80}\right) \simeq 0.987$$

y el número aleatorio r obtenido es 0.886*, que es menor que* 0.987*, por lo que aceptamos este cambio. Seguimos en el siguiente paso con $S = \{f_2, f_3\}$ y quitamos aleatoriamente la variable f_2, obteniendo $S' = \{f_3\}$ que es un peor resultado que S, ya que $J(S) = 1/2 < 1 = J(S')$. Generamos el número aleatorio $r = 0.612$ y la Ecuación* (2.8) *tiene ahora un valor de*

$$\exp\left(\frac{-\Delta J}{T}\right) = \exp\left(\frac{-\left(1 - \frac{1}{2}\right)}{10}\right) = \exp\left(-\frac{1}{20}\right) \simeq 0.9512$$

Por lo que aceptemos este cambio. Además, como es la segunda vez que encontramos un resultado peor descendemos la temperatura $T = 0.4 * 10 = 4$. Ahora, consideramos $S = \{f_3\}$ e introducimos por ejemplo f_1, con lo cual se obtiene un resultado mejor que el anterior, ya que $J(\{f_1, f_3\}) = 0$, por lo que lo aceptamos sin más. Seguimos con el siguiente cambio, esta vez añadiendo f_2 a $S = \{f_1, f_3\}$ obteniendo un resultado igual que el anterior. Es decir, que para $S' = \{f_1, f_2, f_3\}$ tenemos que $J(S) = J(S')$. Al no haber mejoras se evalúa la Ecuación (2.8)*. Sabemos que dicha expresión vale 1, ya que $J(S) = J(S')$ y, por lo tanto, vamos a aceptar el cambio, Este hecho lo comentamos para mencionarlo como un caso de no mejora para el número de iteraciones en que se realizan con temperatura $T = 4$. En la siguiente iteración, eliminamos f_3 resultando $S' = \{f_1, f_2\}$.*

Es decir, tenemos un subconjunto de features *igual al de la iteracion anterior, de modo que se acepta, pero se contabiliza como otra iteración sin mejora. Es decir, ya hemos ejecutado dos iteraciones sin mejora con* $T = 4$ *y volvemos a decender la temperatura. Finalmente,* $T = 4 * 0.4 = 1.6 < 3 = T_1$ *de modo que terminamos el algoritmo obteniendo como resultado final el conjunto* $\{f_1, f_2\}$.

Algoritmos genéticos

Los algoritmos genéticos (*Genetic algorithms*, GA) constituyen otra familia de algoritmos de optimización estocástica. A diferencia del recocido simulado que se inspira en un fenómeno físico, la motivación de los *GA* proviene de la evolución biológica, donde los mejores individuos tienen una mayor probabilidad de sobrevivir. Además, otra diferencia entre SA y GA es que mientras SA solo mantiene un subconjunto de variables en la memoria, los algoritmos genéticos trabajan a la vez con varios subconjuntos de variables o dicho de otra forma, un conjunto de conjuntos de variables. En la terminología GA, los diferentes conjuntos de variables a considerar suelen denominarse cromosomas. Por otro lado, un conjunto de cromosomas se denomina población. El vocabulario biológico se explota aún más cuando definimos operaciones genéticas como mutación o cruce. En la mutación, se añaden o sustraen uno o varias características aleatorias del cromosoma para producir un nuevo cromosoma. Por ejemplo, supongamos que tenemos un conjunto de datos con tres variables $\{f_1, f_2, f_3\}$. Consideremos el cromosoma $(1, 1, 0)$, o lo que es lo mismo, seleccionamos el subconjunto de *features* $\{f_1, f_2\}$. Es decir, que si el valor de la posición p del cromosoma es igual a 1, entonces la variable f_p se incluye en el subconjunto seleccionado. Por el contrario, si en la posición p del cromosoma se incluye un cero, entonces la *feature* f_p no se selecciona. Un posible ejemplo de mutación sería cambiar el primer dígito del cromosoma de 1 a 0, quedando, por lo tanto, $(0, 1, 0)$. Es decir, que sólo seleccionamos la *feature* f_2. Por otro lado, en la operación de cruce, se obtiene un "cromosoma hijo" a partir de dos "cromosomas padres". Dicho de otra forma, tendremos un nuevo subconjunto de variables a través de la combinación de otros dos subconjuntos. Para ello dividiremos los dos cromosomas originales por una de las posiciones aleatoriamente elegidas. A continuación, intercambiaremos las partes divididas para crear nuevos cromosomas. Por ejemplo, supongamos que tenemos los cromosomas $(1, 1, 1)$ y $(0, 0, 0)$. Es decir, en el primer caso, consideramos todas las *features*, mientras que en el segundo no consideramos ninguna. Nótese que los cromosomas elegidos son dos casos extremos que sirven para ilustrar este proceso. El siguiente paso sería elegir una posición por la que dividir. Hemos decidido cortar por el segundo dígito, resultando, por lo tanto, los siguientes cromosomas: $(0, 0, 1)$ y $(1, 1, 0)$. Es decir, en el primer subconjunto $\{f_3\}$, mientras que en el segundo sería $\{f_1, f_2\}$.

Por lo general, una nueva población se forma reteniendo algunos de los cromosomas de la población anterior y componiendo nuevos cromosomas aplicando operaciones genéticas, como la mutación y/o el cruce, en los cromosomas antiguos. Para comparar qué cromosomas son mejores que otros se utiliza una función de evaluación $J(\cdot)$ que mide el ajuste que realiza el modelo entrenado con el conjunto evaluado, es decir, a mayor valor de $J(\cdot)$ mejor es el subconjunto, a diferencia de las funciones de evaluación vistas en métodos anteriores que median el error del modelo. Cuanto mejor es un cromosoma, mayor es su probabilidad de ser seleccionado para la nueva población o como padre en una operación genética. Para una explicación más extensa de esta familia de algoritmos, acudir a [37].

En general, un algoritmo genético consta de cuatro parámetros, N el tamaño de la población, G el número máximo de generaciones (iteraciones), p_c la probabilidad de cruce y

p_m la probabilidad de mutación. El algoritmo comienza con un conjunto de cromosomas de tamaño N, es decir, N subconjuntos aleatorios del conjunto original de variables, P. Cada subconjunto, lo denotamos por S_i, $i = 1, \ldots, N$. En el primer paso seleccionamos cuáles de nuestros cromosomas pasan a la siguiente generación, de modo que los que proporcionan mejores resultados tienen más probabilidad de ser elegidos. Para ello, calculamos el ajuste total de la población, $\mathbf{F} = \sum_{i=1}^{N} J(S_i)$, donde esta vez la interpretación de $J(\cdot)$ es el ajuste del modelo, por lo que cuanto más alto su valor mejor es el resultado, calculamos la probabilidad de selección $p_i = J(S_i)/\mathbf{F}$ y las probabilidades acumuladas, $q_i = \sum_{\ell=1}^{i} p_\ell$ para todo $i = 1, \ldots, n$. Con estas probabilidades, generamos un número aleatorio $r \in [0, 1]$, si $r < q_1$ seleccionamos S_1 para la siguiente generación, en caso contrario elegimos el S_i tal que $q_{i-1} < r \leq q_i$ y repetimos este proceso hasta haber seleccionado N conjuntos. Con este proceso de selección, los conjuntos de características que mejor resultados dan se seleccionan más veces que aquellos que no y los peores desaparecen. Sin embargo, el hecho de poder elegir con probabilidad no nula todos los subconjuntos, nos permite escapar de óptimos locales. A continuación, para cada conjunto de la nueva población generamos un número aleatorio $r_c \in [0, 1]$ y si $r_c < p_c$ lo elegimos para realizar un cruce, juntamos los cromosomas por parejas aleatoriamente, y por cada una se elige una posición aleatoria por donde realizar el cruce. Finalmente, para cada cromosoma elegimos cuáles sufrirán una mutación, generando otro número aleatorio r_m entre 0 y 1 y si $r_m < p_m$ elegir una característica aleatoria para añadir o quitar del cromosoma. Así, terminamos de obtener una nueva población para la siguiente iteración. Repetimos este proceso hasta crear G generaciones.

El pseudocódigo del funcionamiento de los algoritmos genéticos se puede ver en el Algoritmo 4.

A continuación ilustraremos con un ejemplo cómo funcionan los algoritmos genéticos.

Ejemplo 2.12. *Sea nuestra población inicial, el conjunto* $\mathbf{S} = \{S_1 = (1, 0, 1), S_2 = (1, 0, 0), S_3 = (0, 0, 1)\}$, *nuestra base de datos la dada por la Tabla 2.6,* $p_c = 0.7$ *y* $p_m = 0.7$. *En este caso* $J(\cdot)$ *mide el ajuste y no el error cometido por* leave one out *como hemos visto en ejemplos anteriores. El ajuste de los modelos es 1 menos su error cometido por* leave one out. *Ya calculamos los errores cometidos por los conjuntos* $\{f_1, f_2\}$, $\{f_1\}$ *y* $\{f_3\}$ *en los Ejemplos 2.8 y 2.9 que son 0, 0 y 1 respectivamente. Es decir, que el ajuste de los modelos es* $J(S_1) = 1 - 0 = 1$, $J(S_2) = 1 - 0 = 1$ *y* $J(S_3) = 1 - 1 = 0$. *Para esta población inicial tenemos que el ajuste total es* $\mathbf{F} = J(S_1) + J(S_2) + J(S_3) = 1 + 1 + 0 = 2$, *las probabilidades de elección son* $p_1 = J(S_1)/\mathbf{F} = 1/2$, $p_2 = 1/2$ *y* $p_3 = 0/2 = 0$ *y las probabilidades acumuladas son* $q_1 = 1/2$, $q_2 = q_1 + 1/2 = 1/2 + 1/2 = 1$ *y* $q_3 = q_2 + 0 = 1$. *Generamos 3 números aleatorios* $r_1 = 0.709, r_2 = 0.477, r_3 = 0.381$, *como* $q_1 < r_1 \leq q_2$, $r_2 < q_1$ *y* $r_3 < q_1$ *añadimos a la nueva población los conjuntos* S_2 *una vez y el conjunto* S_1 *dos veces obteniendo la nueva población* $\mathbf{S}' = \{(1, 0, 0), (1, 0, 1), (1, 0, 1)\}$, *generamos tres números aleatorios para seleccionar los cruces* $r_{c1} = 0.582, r_{c2} = 0.784, r_{c3} = 0.806$. *Resultando en que solo haríamos un cruce con* $(1, 0, 0)$, *algo imposible porque necesitamos al menos una pareja, por lo que pasamos a la fase de mutaciones. Generamos tres números aleatorios* $r_{m1} = 0.978, r_{m2} = 0.603, r_{m3} = 0.993$, *por tanto, solo vamos a mutar* $(1, 0, 1)$, *elegimos una posición aleatoria, supongamos que es la segunda y la cambiamos. Obteniendo la generación* $\{(1, 0, 0), (1, 1, 1), (1, 0, 1)\}$. *Finalmente, repetir este proceso hasta realizar* G *generaciones.*

Algoritmo 4 *Genetic algorithms*, GA

Función GA(P, J, N, G, p_c, p_m)
Comenzamos con un población aleatorio de tamaño N, $\mathbf{S} = \{S_1, \ldots, S_N\}$ con cada conjunto S_i representado por un vector de unos y ceros.
mientras no se hayan realizado G iteraciones **hacer**
Sea $\mathbf{S}' = \emptyset$ la nueva generación.
Calculamos $\mathbf{F} = \sum_{i=1}^{N} J(S_i)$, $p_i = J(S_i)/\mathbf{F}$ y $q_i = \sum_{\ell=1}^{i} p_\ell$
mientras $\mathbf{S}'$ tenga menos de N elementos **hacer**
Generamos $r \in [0, 1]$
Buscamos i^* tal que $q_{i^*-1} < r \leq q_{i^*}$ y añadimos S_{i^*} a la nueva población $\mathbf{S}'$
terminar mientras
para todo $S_i \in \mathbf{S}'$ **hacer**
Generamos un número aleatorio $r_c \in [0, 1]$
si $r_c < p_c$ **entonces**
Seleccionamos S_i para realizar un cruce.
terminar si
terminar para
Formamos parejas entre todos los seleccionados para hacer cruce.
Realizamos los cruces eligiendo una posición aleatoria por pareja.
para todo $S_i \in \mathbf{S}'$ **hacer**
Generamos un número aleatorio $r_m \in [0, 1]$
si $r_m < p_m$ **entonces**
Realizamos una mutación aleatoria en S_i.
terminar si
terminar para
$\mathbf{S} := \mathbf{S}'$
terminar mientras
devolver S
terminar Función

2.4. Métodos *embedded*

Para concluir este capítulo, analizamos los métodos *embedded*. Estas metodologías seleccionan las variables más representativas y construyen el modelo de clasificación simultáneamente. Desde el punto de vista computacional, tienen la ventaja de ser menos pesados que los *wrapper* [24, 33], ya que no tienen que recalcular el modelo de clasificación en cada paso. Sin embargo, los métodos *embedded* no alcanzan soluciones óptimas como los *wrappers* al no realizar búsquedas tan exhaustivas como estos últimos. Si comparamos los métodos *embedded* con los métodos *filter*, los primeros son computacionalmente más costosos, pero dan mejores resultados de predicción. Entre los métodos *embedded*, podemos encontrar uno especialmente diseñado para SVM conocido como Selección Recursiva de Variables (llamado en inglés, *Recursive Feature Selection*, RFE-SVM) [24, 25]. Esta metodología encuentra un subconjunto de variables de tamaño d de entre las p posibles variables, eliminando aquellas que producen el mayor margen de separación de clases al ser quitadas. Se suelen eliminar en cada iteración la mitad de las variables, ya que eliminarlas de una en una resulta muy ineficiente en altas dimensiones. Otros métodos *embedded* se pueden modelar como problemas de optimización [24, 33, 34, 40]. Esto se hace generalmente incluyendo la selección de variables en el modelo, considerando un término de penalización en la función objetivo, Por ejemplo, a través de la minimización de la "norma cero"(ℓ_0-SVM): $\ell_0(w) = \|w\|_0 = |\{j : w_j \neq 0\}|$. (Se ha escrito "norma cero" con comillas porque realmente no es una norma al no satisfacer la desigualdad triangular).

A continuación, desarrollaremos con más detalle las mencionadas estrategias de selección de variables.

2.4.1. Eliminación Recursiva de Características (RFE-SVM)

La técnica *embedded* conocida como Eliminación Recursiva de Características o en inglés *Recursive Feature Elimination* y denotada abreviadamente RFE-SVM es muy usada para SVM. RFE-SVM es una aplicación del RFE estándar, [25]. RFE estándar se basa en clasificar las características empleando las magnitudes de los pesos w_j de los clasificadores lineales, para ello dada una función objetivo J, que en problemas de clasificación suele medir los errores cometidos, se mide el cambio producido en la función objetivo al eliminar una variable j, denotado por $DJ(j)$ y usarlo como criterio de clasificación. Para clasificadores lineales donde la función objetivo es cuadrática respecto de w_j, como en el caso de SVM ($J(w) = \frac{1}{2}\|w\|^2$), emplear $DJ(j)$ como criterio de clasificación es equivalente a usar w_j^2 en su lugar, [25]. El proceso de eliminación recursiva de características se describe, en general, con el siguiente proceso iterativo:

1. Entrenar el clasificador (en nuestro caso SVM), optimizando los pesos w_j respecto a una función de costes $J(w)$.
2. Calcular el criterio de clasificación para todas las características ($DJ(j)$ o w_j^2).
3. Eliminar la característica con el criterio de clasificación más pequeño.

A continuación, daremos más detalles sobre como aplicar este algoritmo al caso SVM. El método RFE-SVM usa como criterio de clasificación las magnitudes de peso w, correspondientes a las variables de decisión del problema de margen duro (1.8). Es decir, que la función de coste empleada en el paso 1 del RFE general es $J(w) = \frac{1}{2}\|w\|^2$ sujeto a (1.5). Luego, se eleva al cuadrado cada peso w_j. Finalmente, se busca la característica f_j tal que

la correspondiente w_j^2 sea menor y la eliminamos. Se puede encontrar el pseudocódigo de esta metodología en el Algoritmo 5.

Algoritmo 5 RFE-SVM lineal

Función RFE-SVM LINEAL(P, d)
$S := P$
repetir
Calculamos la solución óptima (w^*, b^*) del Problema (1.8).
Buscamos la componente j tal que el valor de $(w_j)^2$ sea menor y eliminamos la correspondiente *feature* f_j.
$S := S \setminus \{f_j\}$.
hasta que solo permanezcan d variables.
devolver S
terminar Función

Este algoritmo puede ser generalizado para eliminar más de una característica por paso. También se puede generalizar el algoritmo al caso no lineal, [24, 25]. Dado que el margen del hiperplano de separación es inversamente proporcional a la norma euclídea del vector de peso w y como hemos visto en el Capítulo 1 por la Ecuación (1.42), esta norma se puede reescribir en términos de las variables duales del modelo SVM:

$$W^2(\alpha) := \sum_{i,\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell K(x_i, x_\ell) = \|w\|^2$$

con $K(x_i, x_\ell) = \langle \phi(x_i), \phi(x_\ell) \rangle$. La característica a eliminar en cada iteración es aquella cuya eliminación minimiza la variación de $W^2(\alpha)$, es decir, $J(w) = W^2(\alpha)$ y se elimina aquella *feature* f_j que minimiza la fórmula $DJ(j) = |W^2(\alpha) - W^2_{(-j)}(\alpha)|$ con $W^2_{(-j)}(\alpha) = \sum_{i,\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell K(x_i^{(-j)}, x_\ell^{(-j)})$ donde la notación $(-j)$ significa que la característica j ha sido eliminada. El RFE-SVM no lineal se describe en el Algoritmo 6.

Algoritmo 6 RFE-SVM no lineal

Función RFE-SVM NO LINEAL(P, d)
$S := P$
repetir
Calculamos la solución óptima α^* del Problema (1.44).
para todo $f_j \in S$ **hacer**
$W^2_{(-j)}(\alpha) = \sum_{i,\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell K(x_i^{(-j)}, x_\ell^{(-j)})$
terminar para
Eliminamos la variable f_{j^*} tal que $j^* = argmin_j(|W^2(\alpha) - W^2_{(-j)}(\alpha)|)$.
$S := S \setminus \{f_{j^*}\}$.
hasta que solo permanezcan d variables.
devolver S
terminar Función

Obsérvese que a diferencia del método SBS visto en Sección 2.3 el algoritmo RFE-SVM no entrena un modelo por cada variable que tiene que evaluar en una iteración, sino que entrena un solo modelo y evalúa el cambio que se produce en la función $W^2(\alpha^*)$ respecto de cada variable para una solución óptima fija α^* en cada iteración.

A continuación, veremos un ejemplo de RFE-SVM para el caso lineal:

Ejemplo 2.13. *Sea $d = 1$ y sean los datos de la Tabla 2.6 nuestra base de datos. Entonces, entrenamos el modelo SVM con kernel lineal con las tres variables que disponemos $S = \{f_1, f_2, f_3\}$. Obteniendo el vector $w^* = (1.336, 0.285, -0.04)$, elevando cada componente al cuadrado tenemos $(w_1^*)^2 = 1.784896, (w_2^*)^2 = 0.081225, (w_3^*)^2 = 0.0016$. Por tanto, como f_3 es la variable cuya componente al cuadrado es más pequeña, la eliminamos del conjunto actual. Seguimos entrenando un modelo con las variables $S = \{f_1, f_2\}$, obtenemos para este caso $w^* = (1.341, 0.291)$ así que $(w_1^*)^2 = 1.798281, (w_2^*)^2 = 0.0846181$ y eliminamos la variable f_2 del conjunto actual. Finalmente, llegamos al conjunto final, $S = \{f_1\}$, que resulta coherente con la intuición que tenemos de usar solo la altura para predecir si una persona es alta o baja.*

2.4.2. Selección de características como problema de minimización

La mayoría de los modelos lineales para selección de características pueden entenderse como el resultado del siguiente problema de minimización:

$$\min_{w,b} \sum_{i=1}^{n} L(\langle w, x_i \rangle + b, y_i) + \lambda \varrho(w) \tag{2.9}$$

donde $L(f(x_i), y_i)$ es una función de errores empíricos, que mide los errores de la función $f(x) = \langle w, x \rangle + b$ en el punto de entrenamiento (x_i, y_i), $\varrho(w)$ es un término de penalización y $\lambda \geq 0$ es un coeficiente de regularización que equilibra el error empírico con este término penalizador. Por lo que, el problema (2.9) se dice tiene forma de '*loss and penalty*', es decir, pérdida y penalización, [24, 34].

Ejemplos comunes de funciones que miden el error empírico tenemos:

- La función perdida *hinge loss* ℓ_{hinge}:

$$\ell_{hinge}(\langle w, x \rangle + b, y) := (1 - y_i(\langle w, x_i \rangle + b))_+$$

  con $(z)_+ := \max\{0, z\}$ la parte positiva de un número z.

- la función de pérdida ℓ_2:

$$\ell_2(\langle w, x \rangle + b, y) := (\langle w, x \rangle + b - y)^2$$

- La función de perdida logística:

$$\ell_{logistic}(\langle w, x \rangle + b, y) := \log(1 + e^{-y(\langle w, x \rangle + b)})$$

Por otro lado, los términos de penalización más usados son:

- La "norma ℓ_0":

$$\varrho(w) = \ell_0(w) := |\{j : w_j \neq 0\}|$$

  es decir, el número de coordenadas distintas de w. Nótese que realmente ℓ_0 no es realmente una norma al no cumplir la desigualdad triangular.

- La norma ℓ_1:

$$\varrho(w) = \ell_1(w) := \sum_{j=1}^{p} |w_j|$$

- La norma ℓ_2:

$$\varrho(w) = \ell_2(w) := \|w\|^2$$

Tomando como función para los errores $L(\langle w, x_i\rangle + b, y) = \ell_{hinge}(\langle w, x\rangle + b, y)$ y sin usar funciones de penalización el problema (2.9) se reescribe uno denominado *Robust Linear Programming* que es muy útil para la generación de hiperplanos separadores, [5]:

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \sum_{i=1}^{n} (1 - y_i(\langle w, x_i\rangle + b))_+ \tag{2.10}$$

A diferencia de los distintos problemas de SVM vistos en Capítulo 1, el Problema (2.10) solo minimiza la media de las observaciones que caen en el lado incorrecto del hiperplano generado, y no se preocupa en encontrar el hiperplano de margen máximo, [5].

A continuación, vamos a agregar distintos términos de penalización $\varrho(w)$ en el problema (2.10) para realizar la selección de variables y mejorar el rendimiento de la generación de los hiperplanos separadores. El nuevo problema con el término de penalización quedaría así:

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \sum_{i=1}^{n} (1 - y_i(\langle w, x_i\rangle + b))_+ + \lambda\varrho(w) \tag{2.11}$$

De hecho, el SVM de margen blando (1.25), visto en Capítulo 1, se puede ver como un problema de pérdida y penalización, (2.9), utilizando como penalización la norma ℓ_2 y como pérdida la norma ℓ_{hinge}, [40]. Tenemos de esa forma el problema:

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \sum_{i=1}^{n} (1 - y_i(\langle w, x_i\rangle + b))_+ + \frac{\lambda}{2}\|w\|^2 \tag{2.12}$$

Ahora, comprobaremos que este problema es equivalente al Problema (1.25) con $\lambda = 1/C$. Si tomamos $\lambda = 1/C$ la función objetivo de (2.12) se trasforma en:

$$\sum_{i=1}^{n} (1 - y_i(\langle w, x_i\rangle + b))_+ + \frac{1}{2C}\|w\|^2 \tag{2.13}$$

Multiplicando la Expresión (2.13) por la constante positiva C no se altera la solución óptima del problema y la función objetivo se reescribe de forma más similar a la del Problema (1.25).

$$C\sum_{i=1}^{n} (1 - y_i(\langle w, x_i\rangle + b))_+ + \frac{1}{2}\|w\|^2$$

Ahora, si denotamos a $(1-y_i(\langle w, x_i\rangle+b))_+$ por ξ_i, vemos que ξ_i cumplen las condiciones del Problema (1.25), $\forall i = 1, \ldots, n$, es decir $\xi_i \geq 0$ e $y_i(\langle x_i, w\rangle + b) \geq 1 - \xi_i$, o lo que es lo mismo,

$$\xi_i \geq \text{máx}(0, 1 - y_i(\langle x_i, w\rangle + b)) = (1 - y_i(\langle w, x_i\rangle + b))_+$$

Sea (w^*, b^*, ξ^*) solución óptima de (1.25) entonces $\xi_i^* = (1 - y_i(\langle w^*, x_i\rangle + b^*))_+$, ya que $\xi_i^* \geq (1-y_i(\langle w^*, x_i\rangle+b^*))_+$ y el término $\sum_{i=1}^n \xi_i$ de la función objetivo de (1.25) al ser suma de valores positivos se minimizara cuando cada ξ_i^* tome el valor más pequeño permitido que es $(1 - y_i(\langle w^*, x_i\rangle + b^*))_+$. Luego (w^*, b^*) es solución de (2.12). Y recíprocamente si (w^*, b^*) es solución óptima de (2.12) entonces $(w^*, b^*, \{(1 - y_i(\langle w^*, x_i\rangle + b^*)_+\}_{i=1}^n)$ es solución de (1.25).

ℓ_1-SVM

Hemos visto en la sección anterior que el problema SVM lineal (1.25) puede verse como un problema de pérdida y penalización, tomando como penalización la norma ℓ_2. Esta norma tiene el efecto de reducir los valores de w, pero no tiene necesariamente que suprimirlos, es decir, hacerse cero. Para este caso, se utiliza como término de penalización la norma ℓ_1. Cuando se utiliza la norma ℓ_1 como término de penalización en (2.11) se obtiene el siguiente problema:

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \sum_{i=1}^{n}(1 - y_i(\langle w, x_i\rangle + b))_+ + \lambda \sum_{j=1}^{p} |w_j| \tag{2.14}$$

Emplear la norma ℓ_1 como penalización cambia en gran medida el resultado final del método, produciendo una selección de características diferente a la que se obtiene con la norma ℓ_2. Esto es debido a la convexidad de la norma ℓ_2, empleada en SVM de margen blando. Para ilustrar este efecto, supongamos que tenemos dos vectores en $\mathbb{R}^2$ dados por $u = (1, 0)$ y $v = (0.5, 0.6)$. Desde el punto de vista del SVM podríamos pensar que u y v podrían tomar el papel de las variables de decisión óptimas w tras resolver el problema de optimización usando la norma ℓ_1 y ℓ_2 respectivamente. En el caso del vector u estaríamos eliminando la segunda variable (por tener valor cero), mientras que con el vector v reducimos los valores de todas las *features*. Pasaremos a calcular la norma ℓ_1 y ℓ_2 de ambos elementos.

$$\begin{aligned} \|u\|_1 &= |1| = 1, & \|v\|_1 &= |0.5| + |0.6| = 1.1, \\ \|u\|_2 &= \sqrt{1^2} = 1, & \|v\|_2 &= \sqrt{0.5^2 + 0.6^2} \simeq 0.78. \end{aligned}$$

Con este ejemplo observamos que $\|u\|_1 < \|v\|_1$, pero $\|u\|_2 > \|v\|_2$. Es decir, que con el uso de la norma ℓ_1 en la penalización de SVM conseguimos eliminar *features*, mientras que con la ℓ_2, eliminaríamos las correspondientes variables solo si los coeficientes se acercan demasiado a cero. Este efecto es lo que se plantea comprobar al emplear la norma ℓ_1, [24].

ℓ_0-SVM

La selección de variables también puede ser abordada usando la mal llamada norma ℓ_0, $\|w\|_0 = |\{j : w_j \neq 0\}|$. Por la naturaleza de la definición de la norma ℓ_0, dados dos vectores u y v con $\|u\|_0 < \|v\|_0$ se tiene que el vector u tiene menos componentes distintas de cero que v, lo que equivale a eliminar más *features* con el vector u que usando el vector v. Al emplear la norma ℓ_0 en el Problema (2.11) como término de penalización se llega al siguiente problema:

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \quad \sum_{i=1}^{n}(1 - y_i(\langle w, x_i\rangle + b))_+ + \lambda\|w\|_0 \tag{2.15}$$

Además, dado que la norma ℓ_0 no es diferenciable, el Problema (2.15) no puede resolverse de forma directa, [24]. Para poder resolverlo en [9] se sugiere aproximar la norma ℓ_0 por la función cóncava:

$$\varrho(w) = \sum_{j=1}^{p} 1 - \exp(-\alpha|w_j|) \approx \|w\|_0 \tag{2.16}$$

con parámetro de aproximación $\alpha > 0$, transformando el Problema (2.15) en:

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \quad \sum_{i=1}^{n}(1 - y_i(\langle w, x_i\rangle + b))_+ + \lambda \sum_{j=1}^{p} 1 - \exp(-\alpha|w_j|) \tag{2.17}$$

El Problema (2.17) se le conoce como *Feature Selection concaVe* (FSV). Para eliminar el valor absoluto de la función objetivo se añade una variable $v \in \mathbb{R}^p$ sujeto a $-v \leq w \leq v$, es decir, $v_j \leq w_j \leq v_j$ para todo $j = 1, \ldots, p$, [9]. Llegando de esta forma a un problema equivalente a (2.17):

$$\begin{aligned} \min_{w\in\mathbb{R}^p, b\in\mathbb{R}, v\in\mathbb{R}^p} \quad & \sum_{i=1}^{n}(1 - y_i(\langle w, x_i\rangle + b))_+ + \lambda \sum_{j=1}^{p} 1 - \exp(-\alpha v_j) \\ \text{s.a.} \quad & -v \leq w \leq v \end{aligned} \tag{2.18}$$

Este problema no es fácil de resolver y se recurre a un algoritmo para hallar las soluciones conocido como Algoritmo de Linealización Sucesiva, conocido en inglés como *Successive Linearization Algorithm* (SLA), introducido en [9].

Combinación de penalizaciones

Dado que la norma ℓ_2 es la responsable de una buena clasificación y las normas ℓ_1 y ℓ_0 de la selección de variables, en [40] sugieren nuevos métodos consistentes en realizar una combinación de estas normas, para lo que necesitaremos dos parámetros de regularización $\mu, \nu \in \mathbb{R}_+$.

Combinando las normas ℓ_1 para la selección de variables y ℓ_2 para la clasificación resolvemos el problema:

$$\begin{aligned} \min_{w\in\mathbb{R}^p, b\in\mathbb{R}, v\in\mathbb{R}^p} \quad & \frac{\mu}{n}\sum_{i=1}^{n}(1 - y_i(\langle w, x_i\rangle + b))_+ + \frac{1}{2}\|w\|^2 + \nu \sum_{j=1}^{p} v_j \\ \text{s.a.} \quad & -v \leq w \leq v \end{aligned} \tag{2.19}$$

Este problema es más sencillo de resolver usando su problema dual al considerar menos variables y tener una estructura similar al SVM.

Por otro lado, considerando las normas ℓ_2 y ℓ_0 y aplicando la aproximación de esta última por (2.16), obtenemos el siguiente problema de minimización:

$$\begin{aligned} \min_{w\in\mathbb{R}^p, b\in\mathbb{R}, v\in\mathbb{R}^p} \quad & \frac{\mu}{n}\sum_{i=1}^{n}(1 - y_i(\langle w, x_i\rangle + b))_+ + \frac{1}{2}\|w\|^2 + \nu \sum_{j=1}^{p} 1 - e^{-\alpha v_j} \\ \text{s.a.} \quad & -v \leq w \leq v \end{aligned} \tag{2.20}$$

En [40], se puede describe un método para resolver el Problema (2.20).

Capítulo 3

SVM y selección de características en R

En este capítulo presentaremos un análisis computacional de los resultados teóricos vistos durante el trabajo, es decir la implementación de SVM y la selección de características. Para ello, usaremos el programa R, [47], con el cual tenemos acceso a una gran diversidad de librerías para varias tareas de análisis de datos, como la clasificación y la selección de características.

Estudiaremos un conjunto de datos de 310 observaciones de pacientes ortopédicos tomados por el Dr. Henrique da Mota, durante un período de residencia médica en el Grupo de Investigación Aplicada en Ortopedia (GARO) de Massues, Lyon, Francia. Los pacientes han sido clasificados en dos categorías: sano y enfermo, donde los pacientes enfermos fueron diagnosticados con Hernia discal o Espondilolistesis. De hecho, disponemos de un total de 100 pacientes sanos y 210 enfermos. Dicha base de datos se puede descargar desde la referencia [3].

A estos pacientes se les ha tomado medidas de 6 características de la columna vertebral:

1. Incidencia pélvica (f_1).
2. Inclinación pélvica (f_2).
3. Ángulo de lordosis lumbar (f_3).
4. Pendiente sacra (f_4).
5. Radio pélvico (f_5).
6. Grado de espondilolistesis (f_6).

Para más información sobre cada una de las variables, consultar [19]. Nuestro objetivo es utilizar la metodología SVM para poder clasificar nuevos pacientes en función de estas 6 características. Para ello, primero analizaremos los resultados de SVM con todas las características en la Sección 3.1. A continuación, en la Sección 3.2, consideraremos varias estrategias de selección de características vistas en el trabajo para conocer cuáles son las *features* más importantes. Con el objetivo de evaluar la eficiencia de la metodología será necesario dividir la muestra original en los subconjuntos de entrenamiento (*train*) y prueba (*test*). Aleatoriamente, hemos tomado el 70 % observaciones para el conjunto *train* y el 30 % para el conjunto *test*. Es decir, el conjunto de entrenamiento estará formado por 217 elementos, mientras que el conjunto de prueba constará de 93 datos. En la Tabla 3.1 podemos ver un resumen del número de elementos en cada muestra, así como el cardinal de los datos de la clase de los pacientes sanos y los enfermos.

	sanos	enfermos	total
original	100 (32 %)	210 (68 %)	310
train	69 (32 %)	148 (68 %)	217
test	31 (33 %)	62 (67 %)	93

Tabla 3.1: Descripción de datos

parámetro C	vectores soporte	errores cometidos	precisión	sensibilidad	especificidad
0.01	139	64	0.656	0.984	0
0.1	103	28	0.828	0.935	0.387
0.5	80	28	0.785	0.839	0.677
1	74	28	0.806	0.855	0.71
2	69	28	0.806	0.855	0.71
5	65	29	0.806	0.855	0.705
50	62	27	0.807	0.855	0.71

Tabla 3.2: Resultados de modelos SVM con kernel lineal. Por orden de izquierda a derecha las columnas representan el valor del parámetro C, el número de vectores soporte del modelo, el número de observaciones de la muestra de entrenamiento que caen en el lado incorrecto del hiperplano definido por el modelo, la precisión con la que se predijeron las observaciones del conjunto *test*, la sensibilidad y especificidad.

3.1. SVM en R

En esta primera sección analizaremos los resultados obtenidos con el modelo SVM con kernel lineal, correspondiente al problema de optimización (1.25). Para ello, usaremos la librería 'e1071'de R, [14], especialmente indicada para resolver problemas de clasificación mediante máquinas de vectores soporte.

Comenzaremos analizando el efecto del parámetro C en el modelo. Con este objetivo, hemos entrenado el SVM lineal para los valores de $C \in \{0.01, 0.1, 0.5, 1, 2, 5, 50\}$ sobre la muestra de entrenamiento de 217 individuos.

A continuación, hemos estudiado la eficacia del SVM en función de varias métricas. En particular, hemos medido el número de vectores soporte obtenidos, el número de errores, así como la precisión, sensibilidad y especificidad. Estas tres últimas características se han medido sobre el conjunto *test*.

El número de errores cometidos por el modelo es el número de observaciones x_i del conjunto *train* que están mal clasificadas, es decir, aquellas observaciones x_i con $\xi_i^* \geq 1$. La precisión se define como la proporción de elementos de la muestra *test* bien clasificados. Por otro lado, la sensibilidad (respect. especificidad) podemos definirla como la proporción de pacientes enfermos (respect. sanos) bien clasificados en el conjunto *test*.

En la Tabla 3.2, podemos observar los resultados obtenidos. Vemos que a medida que aumenta el valor del parámetro C, disminuye el número de vectores soporte que determinan el hiperplano separador y viceversa. Esto es debido a que, tal y como vimos en la Sección 1.2, valores más pequeños de C producen modelos más complejos, es decir, con mayor número de vectores soporte porque permitimos valores más grandes de ξ_i a costa de incluir más ejemplos dentro del margen, incluso, si estos están mal clasificados. Por otro lado, respecto a la precisión, otro aspecto visto en la Sección 1.2 es que valores altos de C producen márgenes pequeños, lo que a su vez produce buenas predicciones y menores valores de C producen un margen más amplio, lo que lleva a realizar peores

predicciones. Por último, en lo que se refiere a la sensibilidad y la especificidad vemos que valores de C bajos mejoran la sensibilidad, mientras que empeora la especificidad. De hecho, en el caso $C = 0.01$, la especificidad es nula, lo que nos indica que todos los pacientes que realmente son sanos se han clasificado como enfermos. Además, para este caso un solo paciente enfermo se ha clasificado como sano. Esto conlleva una precisión de $61/93 = 0.656$ tal y como indica la Tabla 3.2. Una posible explicación a este hecho es que, como hemos visto en la Tabla 3.1, el conjunto de entrenamiento está formado por 148 observaciones de la clase de los enfermos y 69 de observaciones de la clase de los sanos, es decir, hay más del doble de observaciones de la clase de los enfermos que de la clase de los sanos, por lo que se dispone de menos información de la clase de los sanos para modelar. De esta forma, disminuir el valor de C permite que se produzcan más errores. Dado que el Problema (1.25) minimiza la suma de los errores, y tenemos datos desbalanceados (aproximadamente en la proporción 1/3 - 2/3), el modelo cometerá más errores en las observaciones clasificadas como sanas que en las enfermas para intentar minimizar dicha función objetivo.

A continuación, vamos a analizar el comportamiento de la versión no lineal de SVM dado por el problema de optimización (1.44). En particular, tomaremos el kernel radial dado por (1.47). Para ello, hemos entrenado el modelo SVM sobre la muestra de entrenamiento con distintos valores del parámetro de regularización C, y del parámetro del kernel Gaussiano, γ. En particular, hemos tomado los valores, $C \in \{1, 2, 5, 50\}$ y $\gamma \in \{0.01, 0.1, 0.2, 1, 2\}$. La Tabla 3.3 muestra los resultados obtenidos de las métricas anteriormente mencionadas. Además, en la Tabla 3.1, se observa un mapa de calor de la precisión obtenida para diferentes valores de los hiperparámetros, alcanzando el valor máximo en el caso $C = 2$ y $\gamma = 0.01$ y el valor mínimo en el caso $C = 1$ y $\gamma = 0.01$.

En dicha tabla podemos ver que, en general, para C fijo, cuanto mayor es el valor de γ se obtienen más vectores soporte y menos observaciones de entrenamiento acaban en el lado incorrecto del separador. Además, comparando los resultados de la Figura 3.1 con los de la Tabla 3.2 concluimos que, para esta base de datos en particular, es más relevante escoger un kernel apropiado que ajustar la constante C para realizar un mejor ajuste, ya que para un determinado parámetro C la mayoría de los modelos de kernel radial tienen mejor precisión que los de kernel lineal. Por ejemplo, para $C = 1$ solo el modelo con $\gamma = 0.01$ es peor en cuanto a la precisión que el modelo con kernel lineal. Para $C = 2$ y $C = 5$ todos los modelos de kernel radial son mejores, sin importar el valor que tome γ. Finalmente, para $C = 50$ solo el modelo radial con $\gamma = 0.2$ es peor que el modelo con kernel radial.

3.2. Selección de características en R

En esta sección, vamos a aplicar algunos de los métodos de selección de variables a la base de datos elegida.

3.2.1. Evaluación de métodos *filters*

Empezaremos por los métodos *filters*. Para ello, usaremos la librería 'Rdimtools', [51], para evaluar el método de *Fisher Criterion Score*, que aplicado a nuestros datos de estudio nos da el siguiente orden de relevancia de las variables de mayor a menor: f_6, f_5, f_1, f_2, f_3 y f_4. Es decir, que la variable f_6 es la más relevante de acuerdo al *Fisher Criterion Score*. Obsérvese que la variable f_6 está asociada al grado de espondilolistesis y una de las

(C, γ)	vectores soporte	errores cometidos	precisión	sensibilidad	especificidad
(1, 0.01)	135	32	0.785	0.952	0.452
(1, 0.1)	97	24	0.828	0.887	0.71
(1, 0.2)	99	23	0.839	0.903	0.71
(1, 1)	172	14	0.839	0.936	0.645
(1, 2)	206	8	0.817	0.936	0.58
(2, 0.01)	123	24	0.889	0.946	0.768
(2, 0.1)	85	23	0.828	0.871	0.742
(2, 0.2)	90	23	0.85	0.887	0.774
(2, 1)	163	10	0.828	0.919	0.645
(2, 2)	201	2	0.85	0.952	0.645
(5, 0.01)	103	23	0.839	0.903	0.71
(5, 0.1)	75	24	0.823	0.871	0.742
(5, 0.2)	83	21	0.817	0.871	0.71
(5, 1)	151	4	0.839	0.919	0.677
(5, 2)	197	1	0.839	0.936	0.645
(50, 0.01)	72	26	0.828	0.887	0.71
(50, 0.1)	71	20	0.828	0.887	0.71
(50, 0.2)	72	12	0.796	0.839	0.71
(50, 1)	130	0	0.807	0.887	0.645
(50, 2)	190	0	0.839	0.919	0.677

Tabla 3.3: Resultados de modelos SVM con kernel radial. Por orden de izquierda a derecha las columnas representan el valor de los parámetros C y γ, el número de vectores soporte del modelo, las observaciones de la muestra *train* que se encuentran en el lado incorrecto del hiperplano, la precisión con la que se predijeron las observaciones del conjunto *test*, la sensibilidad y especificidad.

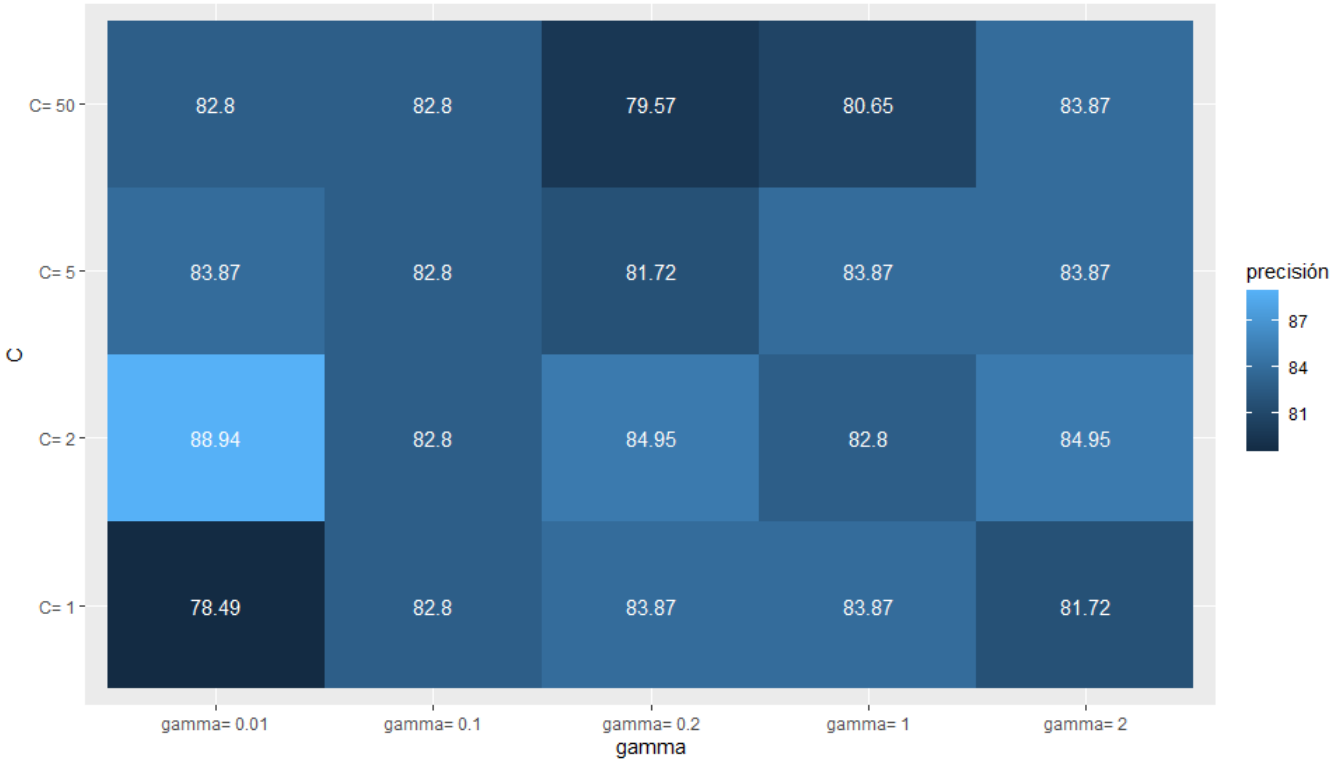


Figura 3.1: Mapa de calor de la precisión de los modelos SVM con kernel radial con distintos valores de C y γ. Cada fila indica un valor de la constante C y cada columna un valor de γ.

enfermedades que nos hace etiquetar a una persona en la clase enferma es precisamente la espondilolistesis, con lo cual parece que los resultados tiene sentido desde el punto de vista práctico.

Para el método *Relief*, visto en la Sección 2.2.2, usaremos librería 'FSinR', [29]. Dado que la selección de las m instancias del método es aleatoria, aplicaremos el método cinco veces con $m = 10$ para obtener una conclusión más consistente. Ordenando las *features* de mayor a menor según el valor de su función de evaluación obtenemos los resultados de la Tabla 3.4. Estos resultados nos indican que las características más relevantes son la f_6 y la f_5, ya que en la mayoría de casos, estas tienen el valor de $J(\cdot)$ más alto. Mientras que, por el contrario, las variables f_1 y f_3 parecen ser las menos relevantes.

$J(f_1)$	$J(f_2)$	$J(f_3)$	$J(f_4)$	$J(f_5)$	$J(f_6)$	conjunto ordenado
0.02	0.038	0.014	-0.015	0.053	0.103	$f_6, f_5, f_2, f_1, f_3, f_4$
0.009	0.001	-0.001	0.019	0.057	0.068	$f_6, f_5, f_4, f_2, f_1, f_3$
0.001	0.04	0.017	0.015	0.045	0.047	$f_6, f_5, f_2, f_3, f_4, f_1$
-0.014	0.046	0.063	0.056	-0.003	0.124	$f_6, f_3, f_4, f_2, f_5, f_1$
0.006	0.074	-0.011	-0.006	0.026	0.081	$f_6, f_2, f_5, f_1, f_4, f_3$

Tabla 3.4: Resultados de método *Relief*. Cada columna es el valor de la función de evaluación de la correspondiente variable, excepto la última que representa las variables ordenadas de mayor a menor por el valor de su función de evaluación.

Para el método CFS usaremos la librería 'FSelector', [43], hallando el subconjunto de atributos óptimo de entre todos los posibles conjuntos. En nuestro caso devuelve solo la

característica f_6.

Vemos que aunque solo haya quedado una variable, esta es f_6 que es coherente con el resultado que daba *Fisher Criterion Score* y *Relief* donde también era la más relevante.

Dado que hemos obtenido en todos los métodos de esta sección que la variable más relevante es f_6, vamos a entrenar un modelo SVM con kernel radial únicamente con la variable f_6 y con los parámetros $C = 1$ y $\gamma = 1/p = 1$ con p igual al número de variables, ya que estos valores son los que se toman por defecto. Obtenemos como resultado que 106 observaciones son vectores soporte, 47 errores en el ajuste del conjunto de entrenamiento, una precisión de 0.774, una sensibilidad de 0.71 y una especificidad de 0.903. Si comparamos con el modelo de la Tabla 3.3 con los mismos parámetros, podemos observar que se ha reducido el número de vectores soporte, es decir, la complejidad del modelo ha disminuido a costa de cometer más errores en el ajuste de los datos de entrenamiento y reducir la precisión. Del mismo modo, también ha disminuido la sensibilidad y ha aumentado la especificidad por el desbalanceo de los datos que ya comentamos previamente. Además, aunque la precisión ha disminuido, el modelo SVM entrenado solo con la variable f_6 tiene mayor precisión que cualquiera de los modelos SVM entrenados con una sola variable, donde esas variables pertenecen al conjunto $\{f_1, \ldots, f_6\}$, ya que ninguno otro supera una precisión de 0.667.

Por otro lado, entrenando el modelo con kernel lineal con $C = 1$ y solamente la variable f_6, el método SVM es equivalente a encontrar un hiperplano en $\mathbb{R}$, lo que corresponde a un punto de la recta real. En la Figura 3.2, vemos la representación en $\mathbb{R}$ de los valores que toma la variable f_6 en el conjunto *train* diferenciando su clase por color, siendo el rojo para los sanos y el azul para los enfermos. En consecuencia, apreciamos que es un conjunto no separable linealmente. Con este experimento, obtenemos 106 vectores soporte, 44 errores en conjunto de entrenamiento, una precisión de 0.774, una sensibilidad de 0.71 y una especificidad de 0.903. Como vemos no hay mucha diferencia entre el modelo con kernel radial y el modelo con kernel lineal empleando únicamente la variable f_6 a pesar de no ser linealmente separable, ya que intuitivamente la zona que le corresponde a cada clase si se puede percibir, porque los pacientes sanos están muy concentrados con unos pocos pacientes enfermos a sus alrededores y viceversa. En la Figura 3.2 la línea vertical continua representa el hiperplano de separación de ambas clases $\langle w, x\rangle + b = 0$ que al estar en $\mathbb{R}$ es equivalente a $w_6 x + b = 0$ y se corresponde con el punto $x = -b/w_6$ y las líneas punteadas a la izquierda y a la derecha corresponden con los márgenes del hiperplano separador $\langle w, x\rangle + b = -1$ y $\langle w, x\rangle + b = 1$, es decir, a los puntos $x = (-1 - b)/w_6$ y $x = (1 - b)/w_6$.

3.2.2. Evaluación de métodos *wrapper*

Utilizando la función *featureSelection* de la librería 'FsinR', [29], podemos realizar la selección de características para métodos *wrapper*. Solo tenemos que especificarle una función de evaluación $J(\cdot)$ y el método de búsqueda seleccionado de los explicados en Sección 2.3.

Durante toda la sección nuestra función de evaluación va a ser un modelo SVM con kernel radial generado mediante la función *wrapperEvaluator*. Vamos a nombrar como ajuste a la proporción de elementos del conjunto *train* bien clasificados. Con el objetivo de visualizar la evolución a través de las distintas iteraciones de los algoritmos SBS y SFS, dibujaremos dicho ajuste, es decir, dicha proporción de puntos, para cada método.

Para realizar el método de búsqueda la selección secuencial hacia atrás (SBS) usamos

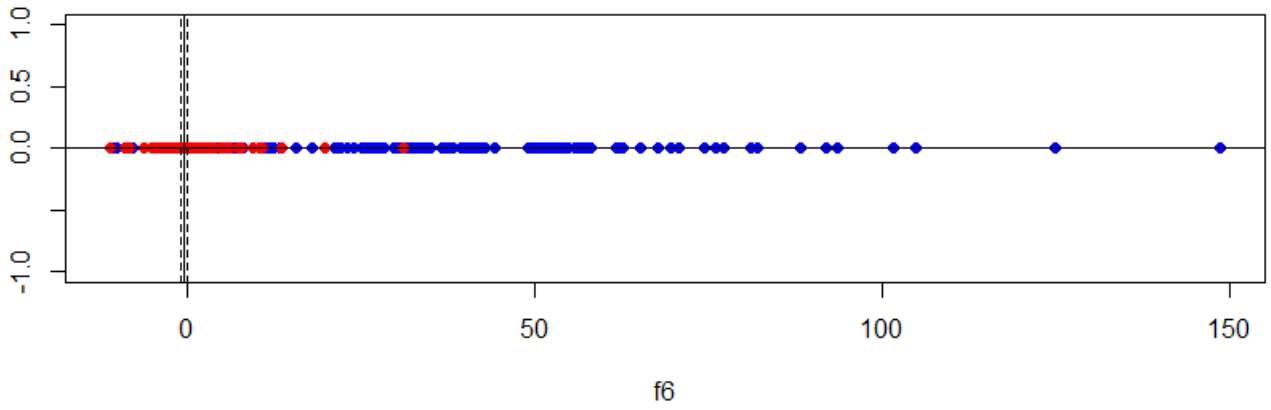


Figura 3.2: Valores de la variable f_6 en el conjunto *train* dibujados en la recta real y el hiperplano de separación lineal y sus márgenes.

la función *sequentialBackwardSelection*, esta función por defecto realiza todas las iteraciones sin ningún tipo de criterio de parada y devuelve el mejor resultado encontrado, es decir, realiza 6 iteraciones, una por cada variable que se elimina del conjunto inicial P. Aplicándolo a nuestro conjunto de entrenamiento obtenemos que las mejores variables son f_2, f_4, f_5 y f_6 con un ajuste de 0.885. En la Figura 3.3 vemos una gráfica de la evolución del mejor ajuste proporcionado, las variables eliminadas fueron f_3 en la primera iteración, f_1 en la segunda, f_2 en la tercera, f_5 en la cuarta, f_4 en la quinta iteración y en la última se eliminó f_6 dejando un conjunto vacío por lo que su ajuste es cero.

Para realizar la selección secuencial hacia adelante (SFS) usamos la función *sequentialForwardSelection* que realiza todo el proceso sin ningún criterio de parada y devuelve el mejor subconjunto que ha encontrado, es decir, realiza 6 iteraciones, una por cada variable que se añade al conjunto inicial vacío. Aplicándolo a nuestra muestra de entrenamiento obtenemos que el mejor conjunto es el formado por f_2, f_3, f_4, f_5 y f_6 con un ajuste de 0.899.

Al establecer como máximo de iteraciones 2 solo se seleccionan las variables f_3 y f_6, con un ajuste de 0.839. Además, al limitar esta función a una iteración se obtiene únicamente la variable f_6, es decir, obtenemos unos resultados coherentes con los vistos en la sección de métodos *filters* (Sección 3.2.1). En la Figura 3.4 vemos una gráfica de la evolución del mejor ajuste proporcionado, las variables añadidas fueron f_6 en la primera iteración, f_3 en la segunda, f_2 en la tercera, f_5 en la cuarta, f_4 en la quinta iteración y en la última se añadió f_1. Obsérvese que en la sexta iteración el ajuste empeora al añadir la variable f_1, ejemplificando el porqué es necesaria la selección de características.

3.2.3. Evaluación de métodos *embedded*

Para implementar los métodos *embedded* usaremos la librería 'mt', [48], para el método RFE-SVM y 'penalizedSVM', [4], para SVM con penalizaciones.

Empleando la función *fs.rfe* en nuestro conjunto de entrenamiento con los parámetros por defecto $C = 1$ y $\gamma = 1/p$ siendo p el número de variables, en nuestro caso $\gamma = 1/6$, y sin establecer en el criterio de parada el número de variables d, obtenemos que las

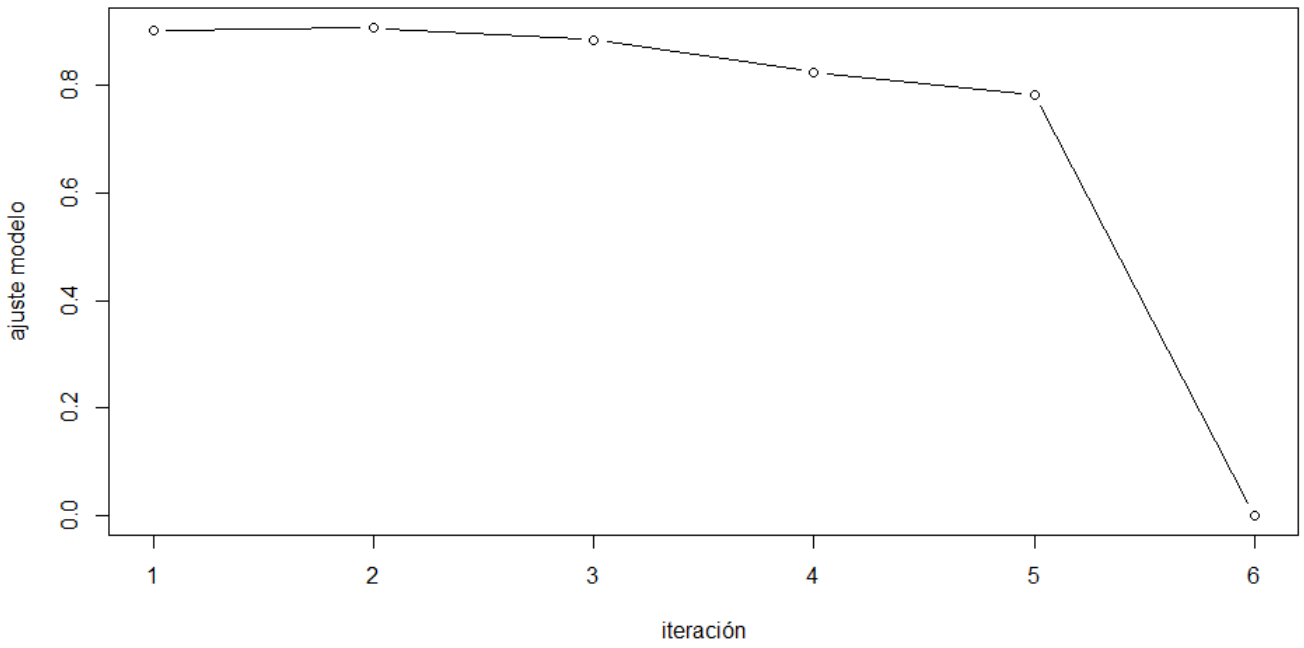


Figura 3.3: Evolución del mejor ajuste proporcionado en el método selección secuencial hacia atrás (SBS).

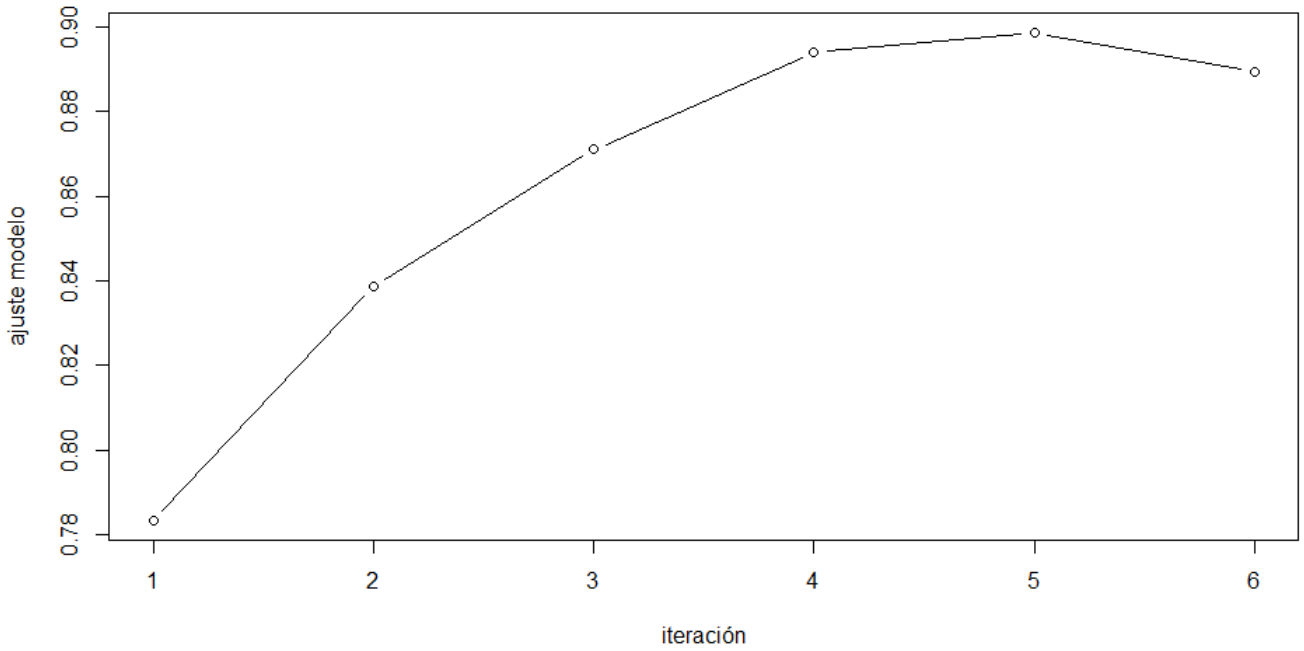


Figura 3.4: Evolución del mejor ajuste proporcionado en el método selección secuencial hacia adelante (SFS).

variables se eliminan en el siguiente orden f_1, f_3, f_2, f_4, f_5 y f_6. Por lo que, el orden de relevancia es el inverso: f_6, f_5, f_4, f_2, f_3 y f_1. Un resultado que encaja con los vistos con anterioridad, donde f_6 y f_5 son las características más relevantes.

Para realizar el modelo ℓ_1-SVM usamos la función *lpsvm* que devuelve que las variables escogidas son f_2, f_3, f_4, f_5 y f_6. Por lo que solo se ha eliminado la variable f_1. Aunque, viendo que los coeficientes obtenidos fueron $w_2 = 0.042$, $w_3 = -0.008$, $w_4 = -0.051$, $w_5 = -0.056$ y $w_6 = 0.078$, el coeficiente de f_3 es el más próximo a cero indicando que es el menos relevante de las que quedan, por lo que se podría eliminar la variable f_3 y entrenar otro modelo.

Capítulo 4

Conclusiones

En este trabajo hemos presentado y desarrollado una de las técnicas de clasificación binaria más populares: las Máquinas de Vectores Soporte (*Support Vector Machine*, SVM). Esta metodología se ha explicado tanto desde su versión más sencilla, para el caso de puntos linealmente separables por un hiperplano, hasta una más compleja, dada por los hiperplanos no lineales.

Respecto a la cuestión de selección de características, hemos descrito las tres principales estrategias presentes en la literatura, destacando las ventajas e inconvenientes de cada una respecto de las otras. También nos hemos centrado en el desarrollo de las más destacables: *Fisher Criterion Score*, *Relief* y CFS para los métodos *filters*; distintas estrategias de búsqueda del conjunto de variables óptimo para los métodos *wrappers* y el método RFE-SVM y una reinterpretación de la Máquina de Vectores Soporte como problema de penalización por parte de las metodologías *embedded*.

Desde el punto de vista práctico, hemos analizado una base de datos real mediante el programa R. Para ello, hemos estudiado la sensibilidad de los distintos hiperparámetros del SVM, así como su comportamiento en función del kernel usado. Para finalizar, hemos estudiado cuáles son las *features* más importantes de acuerdo a las distintas estrategias de selección de variables.

Como futuras líneas de trabajo, se podría proponer el estudio del SVM en problemas de multiclase, [15, 23], regresión, [24, 28] y descripción de datos, [45].

Bibliografía

[1] Emile Aarts, Jan Korst, and Wil Michiels. Simulated annealing. *Search methodologies: introductory tutorials in optimization and decision support techniques*, pages 187–210, 2005.

[2] Noushin Rezapour Asheghi, Katja Markert, and Serge Sharoff. Semi-supervised graph-based genre classification for web pages. In *Proceedings of TextGraphs-9: The workshop on graph-based methods for natural language processing*, pages 39–47, 2014.

[3] Guilherme Barreto and Ajalmar Neto. UCI irvine machine learning repository. https://archive.ics.uci.edu/dataset/212/vertebral+column, 2011.

[4] Natalia Becker, Wiebke Werft, and Axel Benner. Package 'penalizedsvm'. 2010.

[5] Kristin Bennett and Olvi Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, pages 23–34, 1992.

[6] Daniel Berrar. Cross-validation, 2019.

[7] John Adrian Bondy. *Graph theory with applications*. 1982.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[9] Paul Bradley and Olvi Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90, 1998.

[10] Christopher Burges and David Crisp. Uniqueness theorems for kernel methods. *Neurocomputing*, pages 187–220, 2003.

[11] Enrique Carmona. Tutorial sobre máquinas de vectores soporte (SVM). *Disponible en http://www.ia.uned.es/~ejcarmona/publicaciones/%5B2013-Carmona%5D%20SVM.pdf*, pages 1–12, 2014.

[12] Emilio Carrizosa, Cristina Molero-Río, and Dolores Romero Morales. Mathematical optimization in classification and regression trees. *Top*, pages 5–33, 2021.

[13] Emilio Carrizosa and Dolores Romero Morales. Supervised classification and mathematical optimization. *Computers & Operations Research*, pages 150–165, 2013.

[14] Evgenia Dimitriadou, Kurt Hornik, Friedrich Leisch, David Meyer, and Andreas Weingessel. The e1071 package. *Misc Functions of Department of Statistics (e1071), TU Wien*, pages 297–304, 2006.

[15] Kai-Bo Duan and Sathiya Keerthi. Which is the best multiclass SVM method? An empirical study. In *International workshop on multiple classifier systems*, pages 278–285. Springer, 2005.

[16] Richard Duda, Peter Hart, and David Stork. Pattern classification. *John Wiley & Sons*, 2001.

[17] Sulaf Elshaar and Samira Sadaoui. Semi-supervised classification of fraud data in commercial auctions. *Applied Artificial Intelligence*, pages 47–63, 2020.

[18] Usama Fayyad and Keki Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, volume 93, pages 1022–1029. Citeseer, 1993.

[19] Rafael García de Sola. Balance sagital: una introducción teórica. https://neurorgs.net/docencia/sesiones-residentes/balance-sagital-una-introduccion-teorica/, 2018.

[20] Zoubin Ghahramani. Unsupervised learning. In *Summer school on machine learning*, pages 72–112. Springer, 2003.

[21] Sarah Graves, Gregory Asner, Roberta Martin, Christopher Anderson, Matthew Colgan, Leila Kalantari, and Stephanie Bohlman. Tree species abundance predictions in a tropical agricultural landscape with a supervised classification model and imbalanced data. *Remote Sensing*, page 161, 2016.

[22] Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized Fisher score for feature selection. *arXiv preprint arXiv:1202.3725*, 2012.

[23] Husheng Guo and Wenjian Wang. An active learning-based SVM multi-class classification model. *Pattern recognition*, pages 1577–1597, 2015.

[24] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[25] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, pages 389–422, 2002.

[26] Mark Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[27] Mark Hall. Correlation-based feature selection of discrete and numeric class machine learning. 2000.

[28] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[29] Alfonso Jiménez-Vílchez, Francisco Aragón-Royón, Adan Rodriguez, Antonio Arauzo-Azofra, and José Manuel Benítez. Package 'fsinr', 2022.

[30] Asha Gowda Karegowda, A.S. Manjunath, and M.A. Jayaram. Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, pages 271–277, 2010.

[31] Kenji Kira and Larry Rendell. A practical approach to feature selection. In *Machine learning proceedings 1992*, pages 249–256. Elsevier, 1992.

[32] Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with ReliefF. *Applied Intelligence*, page 39 – 55, 1997.

[33] Sebastián Maldonado, Juan Pérez, Richard Weber, and Martine Labbé. Feature selection for support vector machines via mixed integer linear programming. *Information sciences*, pages 163–175, 2014.

[34] Sebastián Maldonado, Richard Weber, and Fazel Famili. Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information sciences*, pages 228–246, 2014.

[35] Olvi Mangasarian. Duality in nonlinear programming. *Quarterly of Applied Mathematics*, pages 300–302, 1962.

[36] Wout Megchelenbrink, Elena Marchiori, and Peter Lucas. Relief-based feature selection in bioinformatics: detecting functional specificity residues from multiple sequence alignments. *Radboud University, Nijmegen*, 2010.

[37] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer Berlin, Heidelberg, 1996.

[38] Hédi Nabli. An overview on the simplex algorithm. *Applied Mathematics and Computation*, pages 479–489, 2009.

[39] Narendra and Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, pages 917–922, 1977.

[40] Julia Neumann, Christoph Schnörr, and Gabriele Steidl. Combined SVM-based feature selection and classification. *Machine learning*, pages 129–150, 2005.

[41] Raul Palma, Luis de Marcos, Daniel Rodriguez, and Amparo Alonso. Distributed correlation-based feature selection in spark. *Information Sciences*, pages 287–299, 2019.

[42] Guadalupe Pérez-Caballero, Jose Manuel Andrade-Garda, P Olmos, Y Molina, I Jiménez, JJ Durán, Carlos Fernández-Lozano, and Floriberto Miguel-Cruz. Authentication of tequilas using pattern recognition and supervised classification. *TrAC Trends in Analytical Chemistry*, pages 117–129, 2017.

[43] Piotr Romanski and Lars Kotthoff. Package 'fselector'. *http://cran/r-project.org/web/packages/FSelector/index.html*, 2013.

[44] Lin Sun, Tianxiang Wang, Weiping Ding, Jiucheng Xu, and Yaojin Lin. Feature selection using Fisher score and multilabel neighborhood rough sets for multilabel classification. *Information Sciences*, pages 887–912, 2021.

[45] David Tax and Robert Duin. Support vector data description. *Machine learning*, pages 45–66, 2004.

[46] Ryan Urbanowicz, Melissa Meeker, William La Cava, Randal Olson, and Jason Moore. Relief-based feature selection: Introduction and review. *Journal of biomedical informatics*, pages 189–203, 2018.

[47] William Venables, David Smith, and R Development Core Team. An introduction to R, 2009.

[48] Lin Wanchang. Package 'mt', 2024.

[49] Philip Wolfe. A duality theorem for non-linear programming. *Quarterly of applied mathematics*, pages 239–244, 1961.

[50] Hujun Yin, Xin Yao, Peter Tino, Emilio Corchado, and Will Byrne. *Intelligent Data Engineering and Automated Learning-IDEAL 2007: 8th International Conference, Birmingham, UK, December 16-19, 2007, Proceedings*, volume 4881. Springer, 2007.

[51] Kisung You, Changhee Suh, and Dennis Shung. Package 'rdimtools', 2022.

[52] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[53] Marinka Zitnik, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael Hoffman. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion*, pages 71–91, 2019.