Support Vector Regression desde una perspectiva multiobjetivo: Una aplicación a la esperanza de vida

Grado en Estadística Empresarial
Facultad Ciencias Sociales y Jurídicas de Elche
Universidad Miguel Hernández

Trabajo Fin de Grado

AUTORA:
Daria Ivanova
TUTORAS:
Lidia Ortiz Henarejos
María Josefa Cánovas Cánovas

2019/2020

UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

Índice general

Biblioteca
UNIVERSITAS Miguel Hernández

Resumen

En el presente proyecto se lleva a cabo una breve recopilación de métodos bien conocidos basados en el aprendizaje automático Support Vector Machine (SVM) y Support Vector Regression (SVR) para la resolución de problemas de clasificación y regresión, respectivamente.

El primer método, a grandes rasgos, consiste en la búsqueda de hiperplanos capaces de separar las diferentes clases. El segundo método, también reside en la búsqueda de un hiperplano regresor que describa los datos con la mayor precisión posible.

El SVR es el sucesor del SVM, es por ello que hay menos estudios y menos literatura del mismo. Sin embargo, tiene una aplicación amplia debido a que la variable respuesta es continua. En este trabajo nos centraremos en la técnica de SVR.

Teniendo en cuenta que el SVR resulta ser un problema de Programación Multiobjetivo (PMO), se propone un planteamiento y una resolución alternativos al existente.

Una vez definidos los diferentes modelos de optimización, se utilizan los lenguajes de programación $\mathcal{R}$ y *MATLAB* para programar los correspondientes algoritmos que permiten resolver estos problemas. Esta implementación, que resuelve diferentes modelos de SVR de forma genérica, constituye una de las principales aportaciones originales del trabajo.

Para ilustrar los diferentes modelos de optimización propuestos, se lleva a cabo una aplicación sobre una base de datos real que nos permitirá realizar, mediante la técnica descrita, predicciones sobre la esperanza de vida de los países de la Organización para la Cooperación y el Desarrollo Económicos (OCDE).

Palabras clave: *Máquinas de Soporte Vectorial, SVM, Regresión de Soporte Vectorial, SVR, optimización, programación lineal, programación cuadrática, programación multiobjetivo, esperanza de vida, modelos predictivos.*

Abstract

In the present project a brief compilation of well known methods based on machine learning is brought off. These methods are Support Vector Machine (SVM) and Support Vector Regression (SVR), for the resolution of classification and regression problems, respectively.

The first method, broadly speaking, consists of the search of hyperplanes capable of separating the different classes. The second method also lies in searching a regressor hyperplane that describes the data as accurately as possible.

The SVR is the successor of the SVM, that is why there are less related studies or literature. Nevertheless, it has a wide application because the response variable is continuous. In this project we will focus on SVR technique.

Considering that the SVR turns out to be a Multi-objective Programming (MOP) problem, an alternative approach and resolution to the existing one is proposed.

Once the different optimization models are defined, $\mathcal{R}$ and *MATLAB* programming languages are used to program the pertintent algorithms that allow these problems to be solved. This implementation, which solves different SVR models in a generic way, constitutes one of the main original contributions of the work.

To illustrate the different optimization models proposed, the algorithms will be applied over a real database, which will allow us to make predictions, using the above-mentioned technique, about the life expectancy in Organisation for Economic Co-operation and Development (OECD) countries.

Key words: *Support Vector Machine, SVM, Support Vector Regression, SVR, optimization, linear programming, quadratic programming, multi-objective programming, life expectancy, predictive models.*

Introducción

Ante el aumento de información disponible en organismos privados y públicos, en nuestra vida cotidiana y en el mundo en general, surgen nuevos métodos de gestión y análisis de esta. Actualmente hay mecanismos de recogida de datos y múltiples técnicas que la sintetizan. Estamos rodeados de datos y el deseo de explotarlos cada vez es mayor, siendo uno de los objetivos principales mejorar y facilitar las tareas ante las que nos enfrentamos en nuestro día a día.

Es importante diferenciar dato e información. Un dato es una representación simbólica que describe alguna situación o conocimiento, sin aportar sentido ni ningún mensaje. En cambio, la información es un conjunto de datos, el cual ha sido procesado adecuadamente y aporta un mensaje que tiene sentido comunicacional y una función social. Cabe destacar que los datos de forma aislada carecen de utilidad. Para más consultas, véase [1].

Uno de los tipos de análisis de datos que más solicitado y de moda se encuentra en la actualidad es el aprendizaje automático o *Machine Learning*. Se observa, cada vez en mayor medida, la necesidad de identificar patrones o tendencias que se esconden detrás de los datos, de predecir y clasificar lo que no conocemos y obtener esta información con cierta rapidez.

El *Machine Learning* se diferencia entre aprendizaje supervisado y no supervisado. En el primer caso, los algoritmos se nutren de unas variables de entrada (*input data*) y devuelven una salida (*output*). Básicamente, el algoritmo se entrena a partir de datos históricos y aprende a predecir sobre nuevos datos. El objetivo puede ser de clasificación o de regresión. En el segundo caso, no se dispone de *outputs* y el fin principal es describir la estructura de los datos y agruparlos según la afinidad de estos. Véase [3] para información adicional.

Este proyecto se centra en el aprendizaje supervisado, concretamente en los métodos SVM y SVR. Se entra en más detalle y se proponen nuevos planteamientos para la segunda técnica.

La estructura del trabajo se compone de tres capítulos y unas conclusiones finales. El capítulo I (1) se basa en una breve recopilación bibliográfica de la técnica SVM y una introducción del modelo de SVR. Seguidamente, el capítulo II (2) versa sobre SVR desde una perspectiva de un problema de optimización multiobjetivo. Por último, el capítulo III (3) se centra en una aplicación a datos reales sobre la esperanza de vida en países de la OCDE.

Es importante señalar que gran parte del esfuerzo que se ha dedicado a la elaboración de este trabajo se centra en la programación de los modelos que se introducen en los capítulos I y II. La principal aportación son los algoritmos que se encuentran a disposición en los anexos (A), (B) y (C).

Objetivos

Los objetivos de este estudio son:

- Describir los diferentes métodos de aprendizaje automático supervisado
- Definir Support Vector Machine y Support Vector Regression
- Definir unos planteamientos alternativos para la resolución de problemas de Support Vector Regression desde una perspectiva de problemas de optimización multiobjetivo
- Programar los algoritmos que resuelven problemas de Support Vector Regression
- Aplicar los algoritmos programados a datos reales sobre la esperanza de vida en países de la OCDE
- Comparar los resultados obtenidos

Biblioteca
UNIVERSITAS Miguel Hernández

1. Capítulo I: Breve revisión bibliográfica sobre Support Vector Regression

1.1. Introducción

En el presente capítulo se va a exponer una breve revisión bibliográfica del SVM y del SVR, en español conocidos como Máquinas de Soporte Vectorial y Regresión de Soporte Vectorial, respectivamente. Ambos consisten en un conjunto de algoritmos de aprendizaje supervisado. El SVM fue introducido inicialmente en los años 90 por Vapnik. La idea original era resolver problemas de clasificación binaria, pero en la actualidad se ha extendido para resolver otros tipos de problemas como son de regresión, agrupamiento y multiclasificación. Tienen una aplicación muy amplia, desde categorización de texto e hipertexto hasta análisis de series temporales. Para más información, véase [10].

Se trata de técnicas que forman parte de la popular disciplina del *Machine Learning* (Aprendizaje Automatizado), tan útil en la actualidad por la cantidad y variedad de datos existentes, como por la necesidad de tomar decisiones en función de estos.

A continuación, se definen los métodos SVM y SVR.

1.2. Antecedentes: Support Vector Machine

El SVM es una de las técnicas más poderosas del aprendizaje automático debido a la sencillez de los modelos y a su robustez a la hora de predecir.

En un primer modelo básico, el SVM consiste en construir un hiperplano en un espacio de dimensionalidad alta que separe las dos clases de un conjunto de datos. Cuanto mayor sea esta separación, mejores predicciones se llevarán a cabo. Es decir, una buena separación entre las clases permitirá una correcta clasificación. Para más información, revisar [6].

Dado un conjunto de datos, este se divide en dos subconjuntos que denominaremos de entrenamiento y testeo. Para la construcción del hiperplano, se utiliza el conjunto de entrenamiento. Y para la validación del modelo, se usan los datos de testeo sobre el modelo ya construido.

En el presente trabajo no se lleva a cabo la *validación cruzada* porque el conjunto de datos es de tamaño reducido. La *validación cruzada de k iteraciones (k-fold cross validation)* consiste en dividir los datos originales en k subconjuntos, tomando uno de ellos como conjunto de testeo, y la unión de los restantes $k-1$ subconjuntos como conjunto de entrenamiento. El proceso se repite k veces, y en cada iteración se toma un conjunto de testeo diferente. Al finalizar las k iteraciones, se calcula el promedio de los resultados de precisión y error obtenidos para cada subconjunto de testeo [7].

En términos de optimización, el modelo SVM busca la máxima separación entre las clases, esto es equivalente a maximizar el margen que existe entre ellas. Por ello, los modelos que emplean la técnica SVM son problemas de optimización. En el presente Trabajo Final de Grado (TFG) se dan por conocidos los conceptos básicos de la teoría de optimización: la estructura de los modelos de optimización

lineal y cuadrática, las condiciones de optimalidad en ambos, el conjunto factible y el conjunto de soluciones.

En esta sección se van a explicar diferentes casos de SVM, comenzando por uno de los modelos particulares más sencillos, para posteriormente llegar a un modelo más genérico.

1.2.1. Caso lineal separable

Es el caso más sencillo y el que menos probabilidad tiene de que se dé en la vida real, pero es la base para obtener el problema de optimización que nos permitirá llegar al modelo más general. Para ello, se definen dos conceptos importantes como son el *hiperplano separador* y el *margen*.

Definición 1.2.1. *Sea* $\mathcal{C} = \{(x_1, y_1), ..., (x_n, y_n), n \in \mathbb{N}\}$*, el conjunto de datos de entrenamiento, donde* $x_i \in \mathbb{R}^d$*,* $d \in \mathbb{N}$*, e* $y_i \in \{+1, -1\}$*,* $i = 1, ..., n$*. Sea* $\mathcal{C}^+ = \{x_i \mid (x_i, y_i) \in \mathcal{C},\ con\ y_i = +1,\quad i = 1, ..., n\}$*, y* $\mathcal{C}^- = \{x_i \mid (x_i, y_i) \in \mathcal{C},\ con\ y_i = -1,\quad i = 1, ..., n\}$*. Llamamos* hiperplano separador *de* C^+ *y* C^- *al hiperplano dado por la ecuación:*

$$\langle \omega, x \rangle + b = 0, \tag{1.1}$$

donde ω *es un vector no nulo de* $\mathbb{R}^d$*, y* $b \in \mathbb{R}$*, verificando las desigualdades:*

$$\begin{aligned} \langle \omega, x_i \rangle + b \geq +1 \qquad & si \quad x_i \in \mathcal{C}^+, \quad i = 1, ..., n, \\ \langle \omega, x_i \rangle + b \leq -1 \qquad & si \quad x_i \in \mathcal{C}^-, \quad i = 1, ..., n, \end{aligned}$$

o lo que es lo mismo,

$$y_i(\langle \omega, x_i \rangle + b) \geq 1, \quad i = 1, ..., n. \tag{1.2}$$

El hiperplano que permite separar las dos clases no suele ser único, como se puede observar en la siguiente imagen.

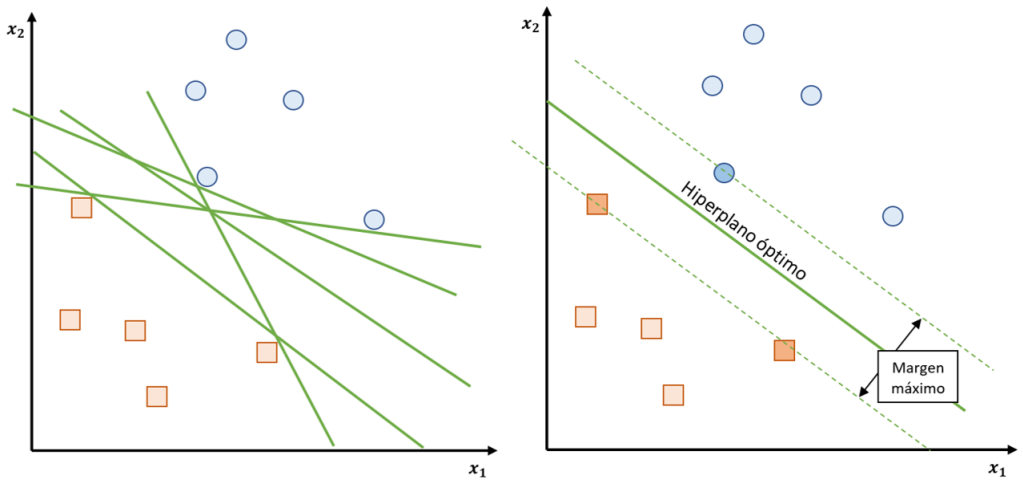


Figura 1.1: Izquierda: posibles hiperplanos de separación entre las dos clases
Derecha: hiperplano que maximiza la separación entre las dos clases.
Fuente propia

Es por ello que se introduce el nuevo concepto de *margen*.

Definición 1.2.2. *Sea el conjunto de datos* $\mathcal{C} = \{(x_1, y_1), ..., (x_n, y_n), n \in \mathbb{N}\}$*, sean los conjuntos* C^+ *y* C^- *introducidos en la definición 1.2.1 y sea* $H = \{x \in \mathbb{R}^d \text{ tal que } \langle \omega, x \rangle + b = 0\}$ *un hiperplano separador de* C^+ *y* C^-*. Dados los hiperplanos* $H^+ = \{x \in \mathbb{R}^d \text{ tal que } \langle \omega, x \rangle + b = +1\}$ *y* $H^- = \{x \in \mathbb{R}^d \text{ tal que } \langle \omega, x \rangle + b = -1\}$*, llamamos* margen *a la distancia entre* H^+ *y* H^-*:*

$$\tau := d(H^+, H^-) = \frac{2}{\|\omega\|^2}. \tag{1.3}$$

Como bien se ha mencionado anteriormente, el objetivo es maximizar el margen (τ), que es equivalente en optimización a minimizar el inverso, $\frac{1}{\tau} = \frac{1}{2}\|\omega\|^2$.

De esta forma, se puede plantear el modelo de optimización final que maximiza el margen sujeto al sistema de desigualdades (1.2).

$$\begin{array}{lccl} (SVM_1) & Min & \frac{1}{2}\|\omega\|^2 & \\ & s.a & y_i(\langle \omega, x_i \rangle + b) \geq 1, & i = 1, ..., n. \end{array} \tag{1.4}$$

(SVM_1) es un problema de optimización cuadrática convexa con restricciones lineales, donde las variables son ω y b, y se resuelve minimizando la norma del vector de pesos ω.

1.2.2. Caso lineal no separable

En la vida real es más probable encontrarnos con problemas que no sean linealmente separables, entonces la misión de encontrar un hiperplano que separe a la perfección los datos se dificulta. Como solución a esta casuística se introduce una nueva variable no negativa, conocida como *variable de holgura* y denotada por $\xi_i, i = 1, ...n$. El objetivo de la misma es hacer que el modelo sea menos rígido y más permisivo con ciertos errores. En otras palabras, recoge los errores cometidos cuando se clasifica de forma incorrecta. Consultar [6] para más información.

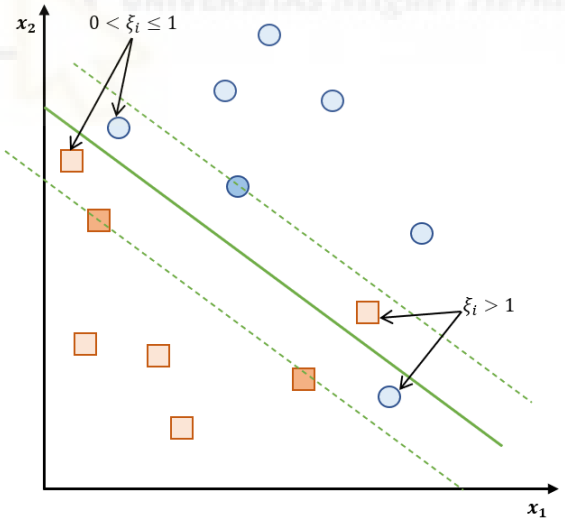


Figura 1.2: Ejemplo caso lineal no separable. Fuente propia

Como es coherente, el problema (1.4) planteado en el apartado anterior varía. Ahora las restricciones tienen la siguiente forma:

$$y_i(\langle \omega, x_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, ..., n. \tag{1.5}$$

En la figura 1.2 se puede ver gráficamente el interés de la variable $\xi_i, i = 1, ...n$, y su significado en función del valor que toma:

- Si $\xi_i = 0 \quad \rightarrow$ Bien clasificado
- Si $0 < \xi_i \leq 1 \quad \rightarrow$ Bien clasificado, pero dentro del margen
- Si $\xi_i > 1 \quad \rightarrow$ Mal clasificado

Para poder controlar los errores $\xi_i, i = 1, ..., n$, se define un nuevo parámetro C, que irá ligado a la suma de estos. Cuanto mayor sea el valor de C, mayor es la penalización que le damos a los errores y por lo tanto, informalmente hablando, estos errores tenderán a decrecer. Por el contrario, cuanto menor sea el valor de C, permitimos que el modelo cometa errores mayores.

El valor de C es importante a la hora de obtener un buen modelo. Un valor demasiado grande supondrá el *overfitting* (sobreaprendizaje), esto significa que el modelo se sobreajusta a los datos de entrenamiento y no se podría generalizar para nuevos datos. Por otro lado, un valor demasiado pequeño implicará el *underfitting*, es decir, el modelo permitirá muchos errores.

Teniendo en cuenta el nuevo parámetro C y las nuevas variables introducidas $\xi_i, i = 1, ...n$, el problema de optimización para SVM, cuando los datos no son linealmente separables, quedaría de la siguiente forma:

$$\begin{array}{lll} (SVM_2) \quad Min & \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\xi_i & \\ s.a & y_i(\langle\omega, x_i\rangle + b) \geq 1 - \xi_i, & \forall i = 1, ..., n, \\ & \xi_i \geq 0, & \forall i = 1, ..., n. \end{array} \tag{1.6}$$

(SVM_2) es un problema de optimización que trata de encontrar el máximo margen permitiendo ciertos errores, donde estos últimos están acotados por su suma y controlados por el parámetro C.

1.2.3. Caso no lineal

En las dos secciones anteriores se han planteado dos modelos que resuelven problemas cuando los datos son separables o casi separables linealmente. No obstante, estos modelos no son directamente aplicables cuando los datos a clasificar no son linealmente separables. En esta sección se aborda cómo adaptar el modelo para este último caso. Para ello, se definirá el *espacio de características* $\mathcal{F}$, con $\mathcal{F} \subset \mathbb{R}^k, k > d$. Para más información, véase [8].

Definición 1.2.3. *Sea* $\Phi : \mathbb{R}^d \rightarrow \mathcal{F}$ *una función de transformación que hace corresponder a cada vector de entrada* $\boldsymbol{x}$ *con un punto en el espacio de características* $\mathcal{F}$ *de mayor dimensión, donde* $\Phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), ..., \phi_k(\boldsymbol{x})]$ *y suponemos que, para algún* $i \in \{1, ..., k\}$, $\phi_i(\boldsymbol{x})$ *es una función no lineal. Llamamos* función de decisión *en el espacio de características a una función de la forma:*

$$D(x) = \langle\omega, \Phi(\boldsymbol{x})\rangle + b, \tag{1.7}$$

donde $\omega \in \mathbb{R}^k$ *y* $b \in \mathbb{R}$.

El modelo resultante final para el caso no lineal es el siguiente:

$$\begin{array}{lll} (SVM_3) \quad Min & \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\xi_i & \\ s.a & y_i(\langle\omega, \phi(x_i)\rangle + b) \geq 1 - \xi_i, & \forall i = 1, ..., n, \\ & \xi_i \geq 0, & \forall i = 1, ..., n. \end{array} \tag{1.8}$$

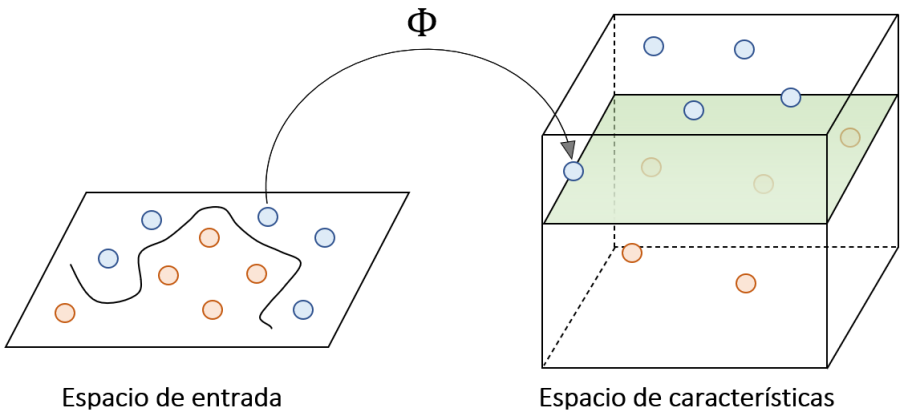


Figura 1.3: Representación gráfica de la transformación de las variables x al espacio de características a través de la función ϕ. Fuente propia

Para la resolución de (SVM_3) existe el concepto de *kernel*. Se trata de una técnica ampliamente utilizada en la cual los vectores de entrada son transformados a un espacio de mayor dimensión donde estos se pueden separar linealmente. De esa forma, el SVM se utiliza para encontrar el hiperplano de margen máximo en el nuevo espacio de características. El hiperplano de separación es una función lineal en el espacio transformado, pero no lineal en el espacio de entrada original. Véase [11].

1.3. Modelo teórico del Support Vector Regression

Dado que el SVM ha demostrado dar buenos resultados, se ha buscado trasladar esa idea a los problemas de regresión, es decir, aquellos en los que la variable respuesta es continua. Por lo tanto, ahora no se dispone de clases para separar. El objetivo en el SVR es seleccionar el hiperplano regresor que mejor se ajuste al conjunto de datos de entrenamiento, considerando una distancia margen ε, de tal modo que se espera que las observaciones se encuentren en una *banda* o *tubo* delimitado por dicho margen.

En los modelos de SVR también se introducen las ya conocidas variables $\xi_i, i = 1, ..., n$, para permitir ciertos errores, acompañadas del parámetro de regularización C en la función objetivo.

El valor de ε, que corresponde a la anchura de la banda, produce un importante impacto en el modelo de manera similar al del valor de C. Si el valor de ε es muy grande, permitimos mucho error y las predicciones del modelo se encontrarán alejadas de los valores experimentales. En cambio, si el valor de ε es muy pequeño, permitimos poco error y el modelo se hace más complejo. Para más información, consultar [6].

A continuación, se presenta el modelo relativo al SVR. En términos de optimización, corresponde a minimizar la mayor de las distancias en vertical, permitiendo que los puntos se encuentren dentro de la franja delimitada por ε con unos errores permitidos ξ_i, ξ_i^*, $i = 1, ..., n$.

$$\begin{array}{lll} (SVR_1) \quad Min & \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) & \\ \quad s.a & y_i - (\langle \omega, x_i \rangle + b) \leq \varepsilon + \xi_i, & i = 1, ..., n, \\ & (\langle \omega, x_i \rangle + b) - y_i \leq \varepsilon + \xi_i^*, & i = 1, ..., n, \\ & \xi_i, \xi_i^* \geq 0, & i = 1, ..., n, \end{array} \tag{1.9}$$

donde:

- d y n son las dimensiones de las variables independientes y el tamaño muestral, respectivamente
- $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}, \quad i = 1, ..., n$, son datos muestrales
- $\varepsilon, \quad C > 0$ son parámetros
- $\omega \in \mathbb{R}^d, \quad b \in \mathbb{R}, \quad \xi = (\xi_i)_{i=1}^n \in \mathbb{R}^n y \xi^* = (\xi_i^*)_{i=1}^n \in \mathbb{R}^n$ son las variables del modelo

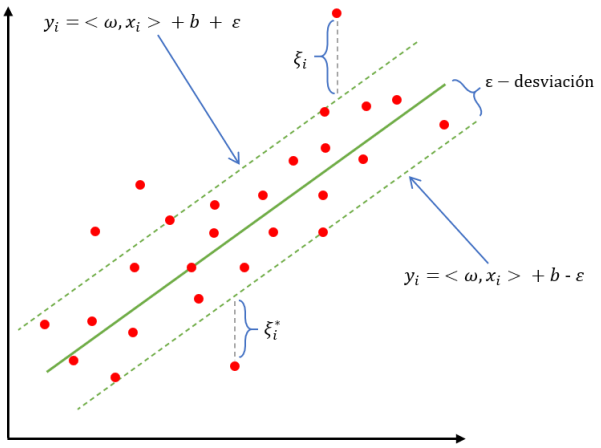


Figura 1.4: Ejemplo de problema Support Vector Regression. Fuente propia

La programación del algoritmo referente al modelo (1.9) introducido se puede consultar en el anexo (A).

2. Capítulo II: Support Vector Regression desde la Programación Multiobjetivo

2.1. Introducción

Como motivación de este capítulo, se puede observar que el modelo (SVR_1) (1.9) coincide con el que resulta de aplicar el método de ponderación de los objetivos a un problema de optimización biobjetivo.

Por motivos de completitud y con el fin de presentar un trabajo autocontenido, a continuación se recuerdan algunos elementos básicos de optimización con varios objetivos involucrados.

2.2. Elementos básicos de la Programación Multiobjetivo

Consideremos el problema de PMO en $\mathbb{R}^p$ dado por:

$$\begin{array}{rll} (PMO) & Min & f_1(x) \\ & Min & f_2(x) \\ & \vdots & \\ & Min & f_q(x) \\ & s.a & g_i(x) \leq 0, \quad i = 1, 2, ..., m, \end{array} \tag{2.1}$$

donde $x \in \mathbb{R}^p$ es la *variable de decisión*, $f = (f_1, ..., f_q) : \mathbb{R}^p \to \mathbb{R}^q$ es la *función objetivo* vectorial de (PMO), y $g_i : \mathbb{R}^p \to \mathbb{R}$, con $i = 1, ..., m$, son las funciones que determinan las *restricciones* del problema (PMO).

Seguidamente, introducimos algunos conceptos básicos relacionados con la resolución del problema de PMO con restricciones.

Definición 2.2.1. *Dado el problema (PMO) introducido en (2.1), se definen los siguientes elementos:*

(i) El conjunto factible *del problema (PMO), denotado por F, es el conjunto de soluciones del sistema de restricciones de (PMO), esto es,*

$$F := \{x \in \mathbb{R}^p \mid g_i(x) \leq 0\,,\ i = 1, ..., m\}\,;$$

(ii) Un punto $\overline{x} \in F$ se dice solución fuertemente minimal *de (PMO) si*

$$f(\overline{x}) \leq f(x)\,,\ \textit{para todo } x \in F;$$

esto es, si $f_j(\overline{x}) \leq f_j(x)$, para todo $x \in F$ y todo $j = 1, ..., q$.

(iii) Un punto $\overline{x} \in F$ se dice solución no dominada *de (PMO) si no existe $y \in F$ tal que*

$$\begin{array}{rcl} f(y) & \leq & f(\overline{x})\,, \\ f_{j_0}(y) & < & f_{j_0}(\overline{x})\,,\ \textit{para algún } j_0 \in \{1, ..., q\}. \end{array}$$

Observación 2.2.1. *En numerosas situaciones, no existen soluciones fuertemente minimales (que optimizan todos los objetivos simultáneamente) para los problemas multiobjetivo, por lo que el concepto de solución no dominada cobra interés. En términos informales, las soluciones no dominadas constituyen soluciones de compromiso entre los diferentes objetivos.*

Podemos encontrar en la literatura diferentes métodos de búsqueda de soluciones no dominadas. Nosotros nos centraremos en dos de estos métodos, que en cierto sentido son complementarios; concretamente, el primero proporciona una condición suficiente, mientras que el segundo una condición necesaria en relación con el concepto de solución no dominada.

Método de las ponderaciones. Consiste en resolver el siguiente problema:

$$\begin{array}{lll} (P_1) & Min & \sum_{j=1}^{q} \alpha_j f_j(x) \\ & s.a & x \in F, \end{array} \tag{2.2}$$

donde $\alpha_j \geq 0$, $j = 1, ..., q$, son los pesos que se atribuyen a los diferentes objetivos.

Proposición 2.2.1. *Con la notación anterior, supongamos que $\alpha_j > 0$, para todo $j = 1, ..., q$. Si $\overline{x}$ es un óptimo global del problema (2.2), entonces $\overline{x}$ es solución no dominada del problema (PMO) introducido en (2.1).*

Método de las restricciones. Consiste en elegir un objetivo como prioritario y acotar el resto. Formalmente, se revuelven problemas del tipo:

$$\begin{array}{lll} (P_2) & Min & f_{j_0}(x) \\ & s.a & f_j(x) \leq b_j,\ j \in \{1, ..., q\}, j \neq j_0, \\ & & x \in F, \end{array} \tag{2.3}$$

eligiendo diferentes valores de las cotas $b_j \in \mathbb{R}$, $j \in \{1, ..., q\}, j \neq j_0$.

Proposición 2.2.2. *Con la notación anterior, si $\overline{x}$ es solución no dominada del problema (PMO), entonces existe un índice $j_0 \in \{1, ..., q\}$ y unas cotas $b_j \in \mathbb{R}$, $j \in \{1, ..., q\}, j \neq j_0$ tales que $\overline{x}$ es un óptimo global del problema (2.3).*

2.3. Modelos teóricos

En primer lugar, obsérvese que el modelo (SVR_1) introducido en (1.9) puede interpretarse como proveniente de aplicar el método de la ponderación de los objetivos al Modelo de Optimización Biobjetivo (MOB) siguiente (véase [5]):

$$\begin{array}{lll} (MOB) & Min & \|\omega\|^2 \\ & Min & \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\ & s.a & y_i - (\langle \omega, x_i \rangle + b) \leq \varepsilon + \xi_i, \quad i = 1, ..., n, \\ & & (\langle \omega, x_i \rangle + b) - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, ..., n, \\ & & \xi_i, \xi_i^* \geq 0, \quad i = 1, ..., n. \end{array} \tag{2.4}$$

Observación 2.3.1. *Nótese que minimizar $\|\omega\|^2$ equivale a maximizar la distancia entre los hiperplanos*

$$\begin{aligned} H^+ &:= \left\{(x, y) \in \mathbb{R}^d \times \mathbb{R} \mid y = \langle \omega, x \rangle + b + \varepsilon\right\}, \\ H^- &:= \left\{(x, y) \in \mathbb{R}^d \times \mathbb{R} \mid y = \langle \omega, x \rangle + b - \varepsilon\right\}, \end{aligned}$$

puesto que dicha distancia en norma euclídea viene dada por

$$dist_2\left(H^+, H^-\right) = \frac{2\varepsilon}{\sqrt{1+\|\omega\|^2}}.$$

Si se considera una norma arbitraria $\|\cdot\|$ *para medir la distancia en* $\mathbb{R}^d \times \mathbb{R}$ *entre* H^+ *y* H^-, *en virtud de la fórmula de Ascoli, tendremos*

$$dist_{\|\cdot\|}\left(H^+, H^-\right) = \frac{2\varepsilon}{\|(1,\omega)\|_*},$$

donde $\|(1,\omega)\|_*$ *representa a la norma dual del vector* $(1,\omega)$. *Por ejemplo, puesto que la norma dual de* $\|\cdot\|_\infty$ *es* $\|\cdot\|_1$, *se tiene que*

$$dist_{\|\cdot\|_\infty}\left(H^+, H^-\right) = \frac{2\varepsilon}{1+\|\omega\|_1}.$$

Observación 2.3.2. *Así pues, el primer objetivo de (MOB) persigue, en términos informales, maximizar la anchura en norma euclídea de la "banda" en torno al hiperplano de ecuación '*$y = \langle\omega, x\rangle + b$*' teniendo en cuenta que se permite una holgura vertical* $\varepsilon > 0$.

Observación 2.3.3. *Si aplicamos el método de ponderación de los objetivos al problema (MOB) con pesos* $\alpha_1 = \frac{1}{2}$ *y* $\alpha_2 = C$, *obtenemos precisamente el modelo* (SVR_1) *presentado en (1.9). Así pues, como consecuencia directa de la proposición 2.2.1 obtenemos el siguiente resultado.*

Corolario 2.3.1. *Si* $(\overline{\omega}, \overline{b}, \overline{\xi}, \overline{\xi}^*) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n$ *es un óptimo global del problema* (SVR_1), *entonces, dicha solución es no dominada para el problema (MOB).*

Se abren entonces diferentes posibilidades, atendiendo al método empleado para encontrar soluciones no dominadas de (MOB). Como alternativa al modelo (SVR_1), surgen de manera natural modelos provenientes de aplicar el método de las restricciones a (MOB). Concretamente, en función del objetivo seleccionado como prioritario, podemos considerar los siguientes modelos:

- Maximizando la anchura de la banda (equivalentemente minimizando $\|\omega\|^2$), teniendo acotada la suma de los errores $(\sum_{i=1}^n (\xi_i + \xi_i^*))$ por una cantidad (nuevo parámetro) $c > 0$:

$$\begin{array}{ll} Min & \|\omega\|^2 \\ s.a & \sum_{i=1}^n (\xi_i + \xi_i^*) \leq c, \\ & y_i - (\langle\omega, x_i\rangle + b) \leq \varepsilon + \xi_i, \quad i = 1, ..., n, \\ & (\langle\omega, x_i\rangle + b) - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, ..., n, \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, ..., n. \end{array}$$

Observación 2.3.4. *Si medimos la distancia entre los hiperplanos* H^+ *y* H^- *en* $\|\cdot\|_\infty$, *de forma natural surgiría el siguiente problema:*

$$\begin{array}{lll} (SVR_2) & Min & \|\omega\|_1 \left(:= \sum_{i=1}^d |\omega_i|\right) \\ & s.a & \sum_{i=1}^n (\xi_i + \xi_i^*) \leq c, \\ & & y_i - (\langle\omega, x_i\rangle + b) \leq \varepsilon + \xi_i, \quad i = 1, ..., n, \\ & & (\langle\omega, x_i\rangle + b) - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, ..., n, \\ & & \xi_i, \xi_i^* \geq 0, \quad i = 1, ..., n, \end{array}$$

que puede resolverse a través de un modelo de Programación Lineal mediante la siguiente transformación estándar:

$$\omega_i := \omega_i^+ - \omega_i^-; \;\; |\omega_i| = \omega_i^+ + \omega_i^-; \;\; con\; \omega_i^+, \omega_i^- \geq 0, \quad i = 1, ..., d.$$

De este modo, aparece el problema:

$$\begin{array}{lll} (SVR_2) & Min & \sum_{i=1}^{d}\left(\omega_i^+ + \omega_i^-\right) \\ & s.a & \sum_{i=1}^{n}\left(\xi_i + \xi_i^*\right) \leq c, \\ & & y_i - \left(\langle \omega^+ - \omega^-, x_i\rangle + b\right) \leq \varepsilon + \xi_i, \quad i = 1, ..., n, \\ & & \left(\langle \omega^+ - \omega^-, x_i\rangle + b\right) - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, ..., n, \\ & & \xi_i, \xi_i^* \geq 0, \quad i = 1, ..., n, \\ & & \omega_i^+, \omega_i^- \geq 0, \quad i = 1, ..., d, \end{array} \tag{2.5}$$

donde las variables son $\omega^+ := \left(\omega_i^+\right)_{i=1}^d \in \mathbb{R}^d$, $\omega^- := \left(\omega_i^-\right)_{i=1}^d \in \mathbb{R}^d$, $b \in \mathbb{R}$, $\xi = (\xi_i)_{i=1}^n \in \mathbb{R}^n$ *y* $\xi^* = (\xi_i^*)_{i=1}^n \in \mathbb{R}^n$. *Y los parámetros en este modelo son* $\varepsilon, c > 0$.

La programación del algoritmo referente al modelo (2.5) se puede consultar en el anexo (B).

- Minimizando la suma de los errores y acotando la anchura de la banda, $\|\omega\|_1 \leq h$, siendo $h > 0$ un nuevo parámetro:

$$\begin{array}{ll} Min & \sum_{i=1}^{n}\left(\xi_i + \xi_i^*\right) \\ s.a & \|\omega\|^2 \leq h, \\ & y_i - \left(\langle \omega, x_i\rangle + b\right) \leq \varepsilon + \xi_i, \quad i = 1, ..., n, \\ & \left(\langle \omega, x_i\rangle + b\right) - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, ..., n, \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, ..., n. \end{array}$$

Si aplicamos de nuevo los diferentes apuntes de la observación 2.3.4, obtenemos el siguiente modelo planteado como un problema de Programación Lineal (PL):

$$\begin{array}{lll} (SVR_3) & Min & \sum_{i=1}^{n}\left(\xi_i + \xi_i^*\right) \\ & s.a & \sum_{i=1}^{d}\left(\omega_i^+ + \omega_i^-\right) \leq h, \\ & & y_i - \left(\langle \omega^+ - \omega^-, x_i\rangle + b\right) \leq \varepsilon + \xi_i, \quad i = 1, ..., n, \\ & & \left(\langle \omega^+ - \omega^-, x_i\rangle + b\right) - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, ..., n, \\ & & \xi_i, \xi_i^* \geq 0, \quad i = 1, ..., n, \\ & & \omega_i^+, \omega_i^- \geq 0, \quad i = 1, ..., d. \end{array} \tag{2.6}$$

En este problema, los parámetros son ε, $h > 0$ y las variables son $\omega^+ := \left(\omega_i^+\right)_{i=1}^d \in \mathbb{R}^d$, $\omega^- := \left(\omega_i^-\right)_{i=1}^d \in \mathbb{R}^d$, $b \in \mathbb{R}$, $\xi = (\xi_i)_{i=1}^n \in \mathbb{R}^n$ y $\xi^* = (\xi_i^*)_{i=1}^n \in \mathbb{R}^n$.

La programación del algoritmo referente al modelo (2.6) se puede consultar en el anexo (C).

Observación 2.3.5. *Como aplicación directa de la proposición 2.2.2, cualquier solución no dominada del problema biobjetivo*

$$\begin{array}{lll} (MOB) & Min & \|\omega\|_1 \\ & Min & \sum_{i=1}^{n}\left(\xi_i + \xi_i^*\right) \\ & s.a & y_i - \left(\langle \omega, x_i\rangle + b\right)) \leq \varepsilon + \xi_i, \quad i = 1, ..., n, \\ & & \left(\langle \omega, x_i\rangle + b\right)) - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, ..., n, \\ & & \xi_i, \xi_i^* \geq 0, \quad i = 1, ..., n, \end{array}$$

ha de ser una solución óptima del problema de (SVR_2) *para algún valor de* $c > 0$, *o de* (SVR_3) *para algún* $h > 0$.

3. Capítulo III: Aplicación: ¿es posible predecir la esperanza de vida?

3.1. Introducción

A lo largo de las últimas décadas se ha observado que la esperanza de vida media en el nacimiento de diferentes países del mundo ha aumentado considerablemente, llegando a duplicarse en países como España, pues si en 1910 la esperanza de vida era de 41.73 años[1], en el año 2017 era de 83.5 años[2]. Se trata de una diferencia abismal, y hay varios motivos que pueden explicar este hecho. Influye el factor genético, los factores ambientales, el sistema de salud pública, la dieta, el estilo de vida, avances de la ciencia y la tecnología aplicados a la sociedad del bienestar, entre otras...

Este aumento de la esperanza de vida tiene grandes consecuencias y repercusiones en el desarrollo del país, pues afecta en la planificación de la jubilación, los costos de la salud, los modelos de trabajo y estrategias de educación [2].

Diversos estudios recientes se han dedicado a estudiar los determinantes de la esperanza de vida en distintos países del mundo. Las conclusiones a las que han llegado es que diferentes factores como el ingreso per cápita, la educación, el acceso a la salud, el gasto per cápita en salud, el número de médicos, la ubicación geográfica, entre otros, son determinantes de la esperanza de vida. En otras palabras, la mejora en las condiciones y calidad de vida de los ciudadanos implica un aumento en la esperanza de vida media del país. Para más información, véase [4].

Para la aplicación sobre datos reales del presente trabajo, se han utilizado como variables explicativas el número de médicos por cada 1000 habitantes, el número de camas por cada 1000 habitantes y el gasto en salud per cápita (US$), con la finalidad de predecir la esperanza de vida en años, donde cada registro son países pertenecientes a la OCDE. Se puede observar que las variables elegidas como independientes son a nivel agregado por país.

Los tres modelos que se programan para la aplicación son los descritos en los capítulos I y II: $(SVR_1)(1.9)$, $(SVR_2)(2.5)$ y $(SVR_3)(2.6)$. Los algoritmos y la descripción de los mismos se pueden consultar en los anexos (A), (B) y (C), respectivamente.

3.2. Banco de datos, variables y software

En el presente apartado, se presenta el banco de datos con el que se trabaja a lo largo de este capítulo, las variables que intervienen y el software que se utiliza para la aplicación y resolución de los problemas introducidos en capítulos anteriores.

Los datos utilizados en el presente estudio han sido obtenidos a partir del banco de datos de libre acceso del *Banco Mundial*[3], que es una organización multinacional que sirve de asistencia financiera y

[1] INE: Instituto Nacional de Estadística
[2] Expansión / Datosmacro.com
[3] https://datos.bancomundial.org

técnica para países en desarrollo. Los registros corresponden a los países miembros de la *Organización para la Cooperación y el Desarrollo Económicos*[4] (OCDE) (excepto aquellos países con campos vacíos en alguna o algunas de las variables) y están datados en 2012.

Variable	Descripción	Unidades
`Esperanza de vida` *(esp_vida)*	La esperanza de vida al nacer indica el número de años que viviría un recién nacido si los patrones de mortalidad prevalecientes en el momento de su nacimiento fueran los mismos durante toda su vida.	Años
`Número médicos` *(num_medicos)*	Los médicos incluyen a médicos de cabecera y especialistas.	Por cada 1000 habitantes
`Número camas` *(num_camas)*	Cantidad de camas disponibles para ingresos en hospitales públicos, privados, generales y especializados, y centros de rehabilitación.	Por cada 1000 habitantes
`Gasto en salud` *(gasto_salud)*	Gastos corrientes en salud per cápita en dólares estadounidenses actuales. Las estimaciones de los gastos de salud actuales incluyen bienes y servicios de salud consumidos durante cada año.	Dólares americanos (US$) / Per cápita

Tabla 3.1: Variables del banco de datos

En la tabla 3.1 se muestran las variables elegidas para la aplicación, junto a su descripción y unidades. Como se ha comentado en la introducción del presente capítulo, diferentes factores influyen en la esperanza de vida, sin embargo, existen ciertos factores que podrían explicar una mayor (o menor) esperanza de vida que tienen que ver con el comportamiento individual, difíciles de detectar con datos de naturaleza agregada [4]. Es por ello que se han elegido variables que recogen los datos agregados a nivel de países.

Para la programación de los algoritmos y resolución de los problemas que se plantean en los capítulos I y II, se utilizan las herramientas $\mathcal{R}$ y *MATLAB*.

$\mathcal{R}$[5] es un entorno de software libre y lenguaje de programación diseñado para el análisis estadístico, que proporciona una amplia variedad de técnicas y herramientas estadísticas y gráficas. *MATLAB*[6] es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio. Ofrece la posibilidad de hacer cálculos numéricos, representar gráficos, implementar algoritmos, etc.

Para programar el modelo cuadrático (SVR_1) se utiliza la función *quadprog* de *MATLAB*, y para los modelos lineales (SVR_2) y (SVR_3) la función *lp* del paquete 'lpSolve' de $\mathcal{R}$.

Es preciso mencionar que existe un paquete en $\mathcal{R}$ llamado 'e1071' que contiene la función *svm*, la cual resuelve problemas de SVM y SVR, entre otros. En el presente trabajo se comparan los resultados del modelo programado (SVR_1) (véase (A)) con los resultados aplicando la función *svm*. El planteamiento de un modelo y el otro difiere, porque el programado en la función *svm* utiliza el concepto de *kernel* y se implementa el dual del problema de optimización primal (véase [9]). Es por ello que en el trabajo aquí presente se decide programar el modelo de optimización cuadrática básico introducido en (1.9).

[4] www.oecd.org
[5] www.r-project.org/about.html
[6] https://es.mathworks.com/products/matlab.html

3.3. Análisis descriptivo

En esta sección se realiza un breve análisis descriptivo de las variables involucradas en la aplicación sobre los modelos. En la tabla 3.1 se puede consultar la definición y las unidades de cada variable.

En primer lugar, en la tabla 3.2 se muestra el conjunto de datos ($n = 31$) ordenado por países, y dos tablas con los tres países con mayor y menor esperanza de vida media.

Países	**Núm. medicos**	**Núm. camas**	**Gasto salud**	**Esperanza de vida**
Alemania	3.92	8.20	4750.42	80.54
Austria	4.85	7.70	4966.38	80.94
Bélgica	2.94	6.30	4543.08	80.39
Canadá	2.40	2.70	5352.77	81.65
Colombia	1.74	1.50	448.23	75.88
Corea	2.08	10.30	1575.79	80.82
Eslovenia	2.54	4.50	1974.97	80.12
España	3.82	3.00	2589.85	82.43
Estados Unidos	2.50	2.90	8438.34	78.74
Estonia	3.28	5.50	1013.32	76.33
Finlandia	3.07	5.30	4406.25	80.63
Francia	3.17	6.50	4650.34	81.97
Grecia	6.08	4.40	1968.40	80.63
Hungría	3.10	7.00	961.14	75.06
Irlanda	2.66	2.80	5294.75	80.85
Islandia	3.51	3.30	3775.84	82.92
Israel	3.34	3.10	2386.76	81.70
Italia	3.86	3.40	3125.61	82.24
Japón	2.27	13.40	5212.07	83.10
Letonia	3.10	5.90	745.52	73.78
Lituania	4.15	7.40	896.96	73.86
Luxemburgo	2.78	5.20	7096.70	81.39
México	2.05	1.50	562.93	74.97
Noruega	4.24	4.00	8970.12	81.45
Polonia	2.22	6.50	815.14	76.75
Portugal	4.08	3.40	1918.60	80.37
Reino Unido	2.73	2.80	3496.26	80.90
República Checa	3.65	6.70	1387.41	78.08
Suecia	4.04	2.60	6273.51	81.70
Suiza	3.90	4.80	9286.55	82.70
Turquía	1.74	2.70	524.82	75.37

Tabla 3.2: Banco de datos.
Fecha: 2012. Fuente: Banco Mundial

Países	**Núm. médicos**	**Núm. camas**	**Gasto salud**	**Esp. de vida**
Japón	2.274	13.4	5212.069	83.096
Islandia	3.508	3.3	3775.8363	82.917
Suiza	3.8987	4.8	9286.55	82.698

Tabla 3.3: Los 3 países con mayor esperanza de vida del banco de datos

Países	**Núm. médicos**	**Núm. camas**	**Gasto salud**	**Esp. de vida**
México	2.049	1.5	562.929	74.966
Lituania	4.149	7.4	896.958	73.863
Letonia	3.095	5.9	745.517	73.778

Tabla 3.4: Los 3 países con menor esperanza de vida del banco de datos

A continuación, presentamos los estadísticos descriptivos para cada variable.

Variable	Mín.	Q1	Mediana	Media	Q3	Máx.
Número médicos	1.74	2.521	3.105	3.219	3.88	6.08
Número camas	1.5	2.95	4.5	5.01	6.5	13.4
Gasto salud	448.2	1200.4	3125.6	3529.3	5089.2	9286.5
Esperanza de vida	73.78	77.41	80.63	79.62	81.68	83.1

Tabla 3.5: Estadísticos descriptivos de las variables

A primera vista, destaca que la diferencia entre el valor mínimo y máximo de la variable 'Esperanza de vida' es de casi diez unidades, o lo que es lo mismo, 10 años. Del mismo modo, la diferencia en el caso de la variable 'Gasto salud' es notablemente grande.

Por otro lado, la mediana y la media de todas las variables no difieren en exceso. Esto implica que aparentemente hay ausencia de valores atípicos.

Por último, se muestra el gráfico 3.1, realizado con la función *chart.Correlation()* del paquete 'PerformanceAnalytics' de $\mathcal{R}$. Nos informa de la correlación entre las variables, el diagrama de dispersión para cada par de variables y el diagrama de barras para cada variable.

Se puede destacar la siguiente información:

- Las variables 'Esperanza de vida' y 'Gasto salud per cápita' están correladas significativamente y de manera positiva
- Si nos fijamos en el gráfico de dispersión de las variables 'Esperanza de vida' y 'Gasto salud per cápita', se puede observar que una vez se llega a un gasto en salud per cápita de aproximadamente 3000 US$, la esperanza de vida se mantiene prácticamente constante.

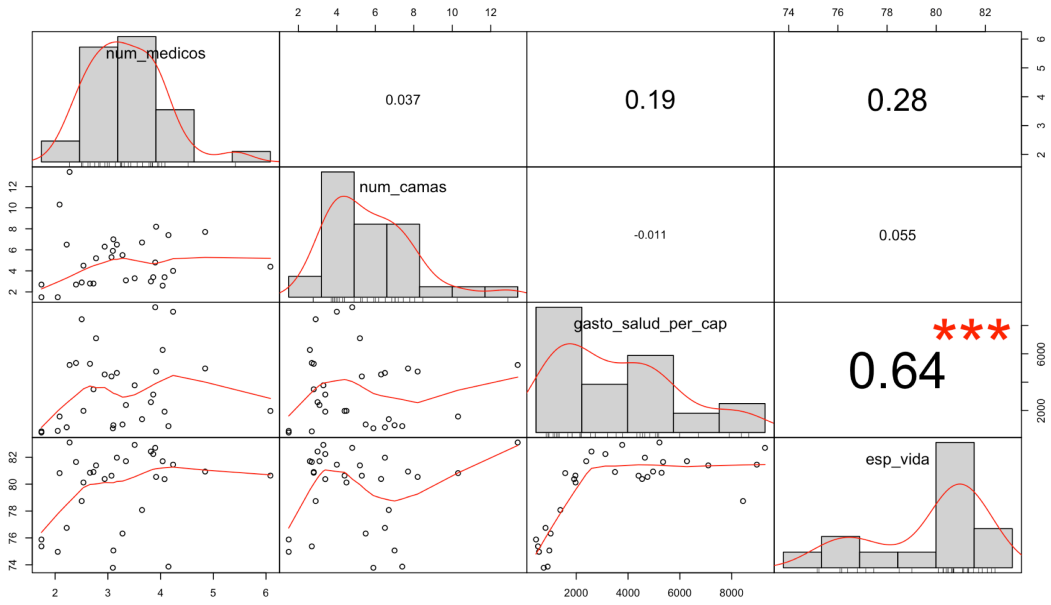


Figura 3.1: Figura: correlaciones, histogramas y diagramas de dispersión

3.4. Resultados

En este apartado se evalúan los modelos que se han definido en los capítulos I y II. Se recuerda que se trataban de problemas de optimización cuadrática para el caso de (SVR_1), y lineal para los casos de (SVR_2) y (SVR_3). La programación de cada uno de ellos se encuentra en los anexos (A), (B) y (C), respectivamente.

Para la evaluación de los modelos, se define una medida de bondad de ajuste que denominaremos como MBA. Dado que la norma que se ha utilizado en los modelos es la norma 1, el cálculo de esta medida se define con valores absolutos. Además, por homogeneidad, se utiliza la misma medida para los tres modelos.

$$MBA = \left(\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|\right)\frac{1}{\frac{1}{n}\sum_{i=1}^{n}|y_i - \overline{y}|}, \tag{3.1}$$

donde n es el tamaño del subconjunto de testeo; $y_i, i = 1, ..., n$, los datos observados de testeo; $\hat{y}_i, i = 1, ..., n$, los valores estimados y $\overline{y}$ la media de los datos observados de testeo.

La medida $MBA \in [0, +\infty)$. Si $MBA \in [0, 1)$, se recomienda utilizar el modelo para predecir antes que usar la media de los valores observados. Cuanto más cercano esté MBA al valor cero, menos error se estará cometiendo y la predicción será más precisa. De esta forma, nos quedaremos con aquel modelo que menor valor de MBA tenga. En cambio, si MBA toma valores por encima de uno o altos, implica que el modelo es menos recomendable que predecir con la media.

Por último, pero no menos importante, se define el orden de las variables para la resolución del problema de optimización (SVR_1):

$$x = (\omega_1, \quad ..., \quad \omega_d, \quad b, \quad \xi_1, \quad ..., \quad \xi_n, \quad \xi_1^*, \quad ..., \quad \xi_n^*),$$

y los problemas de optimización (SVR_2) y (SVR_3):

$$x = \left(\omega_1^+, \quad ..., \quad \omega_d^+, \quad w_1^-, \quad ..., \quad w_d^-, \quad b^+, \quad b^-, \quad \xi_1, \quad ..., \quad \xi_n, \quad \xi_1^*, \quad ..., \quad \xi_n^*\right).$$

- **MODELO CUADRÁTICO** (SVR_1)

Para programar el modelo (1.9) se ha utilizado *MATLAB* y la función *quadprog* que forma parte de dicha herramienta. A continuación, se describen los pasos a seguir en la programación para estimar el modelo. Para consultar el código y los comentarios al detalle véase el Anexo (A).

En primer lugar, se divide el conjunto de datos aleatoriamente en dos subconjuntos denominados entrenamiento (80%) y testeo (20%). El modelo se construye con los datos de entrenamiento.

Para la construcción del modelo, se definen los diferentes vectores y matrices que intervienen en la función objetivo, restricciones y cotas de no negatividad. Se define el parámetro $\varepsilon = 0.5$, que es el error permitido.

Se programa un bucle para guardar las soluciones válidas del problema de optimización para diferentes valores de C. Para ello, se define la siguiente condición de parada: si se cumple que $|\sum(\omega_i) - \sum(\omega_{i-1})| < 0.000001$, para la iteración i del bucle, se para el almacenamiento, ya que a partir de esa iteración, las soluciones prácticamente son similares entre ellas.

De esta forma se obtiene una matriz de resultados *solution*, para diferentes valores de C, con los valores estimados de las variables. Ilustramos los resultados para el valor más pequeño y más grande de C.

Caso 1. Hiperplano para $C = 0.01$:

$$\hat{y} = 0.051 * num\_medicos + 0.0892 * num\_camas + 0.0009 * gasto\_salud + 75.817$$

Tras aplicar el hiperplano a los datos de testeo, obtenemos $\hat{y}$, los valores de y estimados, y posteriormente calculamos la medida MBA a partir de la fórmula introducida en (3.1).

$$MBA_{(C=0.01)} = 0.9383$$

Caso 2. Hiperplano para $C = 0.1$:

$$\hat{y} = 0.3271 * num\_medicos + 0.1251 * num\_camas + 0.0007 * gasto\_salud + 75.301$$

$$MBA_{(C=0.1)} = 0.8942$$

Para ambos casos el valor $MBA < 1$, que indica que el modelo es más recomendable para predecir que utilizar la media. Además, se puede observar que para un valor mayor de C, se tiene un menor valor de la medida MBA. Esto es debido a que a mayor valor de C, menor es la suma de los errores ξ_i, ξ_i^*.

Aparentemente el coeficiente que acompaña a la variable *gasto_salud* es pequeño, pero esto es debido a las unidades. Un aumento en pocos US$ no afectará significativamente a la esperanza de vida. En cambio, un aumento de miles de US$ sí tendría un impacto notorio.

Adicionalmente, con el fin de realizar un pequeño análisis de robustez del modelo (SVR_1), se realiza una comparativa entre el modelo programado y la función *svm* del paquete 'e1071' implementada en $\mathcal{R}$, aplicamos dicha función a los datos. Los pasos a seguir en la estimación mediante el paquete 'e1071' son los mismos: dividir los datos en entrenamiento y testeo, aplicar la función a los datos de

entrenamiento (con kernel lineal), predecir sobre los datos de testeo y calcular la medida MBA. Se obtienen los siguientes resultados:

$$MBA_{(C=0.01)} = 0.9697,$$
$$MBA_{(C=0.1)} = 0.8842.$$

Nuevamente, a mayor valor de C, peor es la precisión del modelo. Además, los valores de MBA no difieren en exceso de los obtenidos con el modelo programado en (A).

- **MODELO LINEAL** (SVR_2)

La programación del modelo (2.5) se lleva a cabo en $\mathcal{R}$ dando uso de la función *lp* del paquete 'lpSolve'. Se puede consultar el código y la explicación en el anexo (B), o clicando en el enlace [7].

Los pasos que se siguen para la construcción del modelo son los mismos que para (SVR_1), a diferencia de las condiciones de comienzo y parada, las cuales están especificadas en el propio anexo (B).

De igual forma, obtenemos una matriz $solution\_fin$ con las soluciones para los distintos valores del parámetro c. En el siguiente gráfico 3.2, podemos observar cómo se comporta $\sum_{i=1}^{d} |\omega_i|$ conforme aumentan los valores de c:

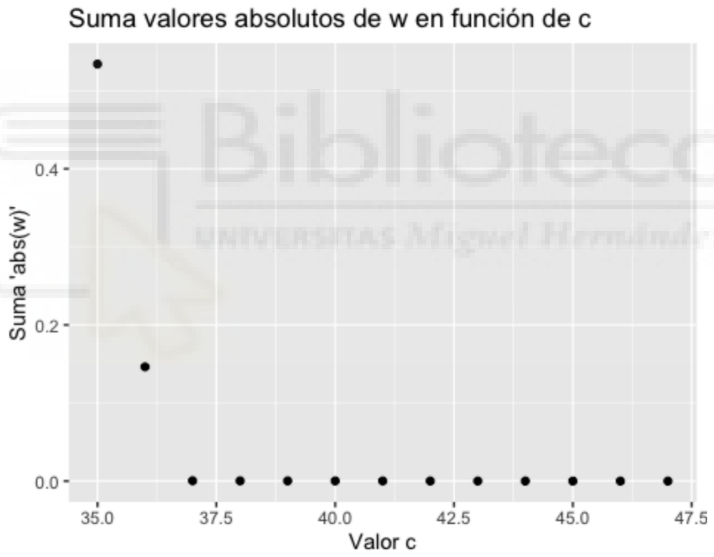


Figura 3.2: Suma valores absolutos de w en función de c

En términos informales, cuanto mayor es el valor de c, menor es la suma de los valores absolutos de ω, es decir, a mayor valor de c, el tubo que envuelve los puntos tiende a colocarse de forma horizontal.

[7] rpubs.com/DashaIvanova/SVR2

Mostramos los resultados para el valor más pequeño y más grande de c.

Caso 1. Hiperplano para $c = 35$:

$$\hat{y} = 0.512 * num\_medicos + 0.0216 * num\_camas + 0.0006 * gasto\_salud + 75.7$$

$$MBA_{(c=35)} = 0.57$$

Caso 2. Hiperplano para $c = 40$:

$$\hat{y} = 0 * num\_medicos + 0 * num\_camas + 0.0002 * gasto\_salud + 79.8$$

$$MBA_{(c=40)} = 0.913$$

Nuevamente, para ambos casos $MBA < 1$. Por otro lado, a mayor valor de c, mayor es el valor de MBA, implicando peor precisión por parte del modelo a la hora de predecir. Se debe a que, cuanto mayor sea el valor de c, se permite que la suma de los errores ξ_i, ξ_i^* también sea mayor. Por último, se observa que, para $c = 40$, los coeficientes que acompañan a las variables *num_medicos* y *num_camas* son nulos, por lo tanto, una variación en esas variables no afectaría a la esperanza de vida.

- **MODELO LINEAL** (SVR_3)

La programación del último modelo (2.6) es muy similar a (SVR_2), por lo tanto, los pasos para su resolución son prácticamente los mismos. Se puede consultar el código y los comentarios en el anexo (C), o clicando en el enlace [8].

Por último, mostramos los resultados para el valor más pequeño y más grande de h.

Caso 1. Hiperplano para $h = 0.1$:

$$\hat{y} = 0 * num\_medicos + 0.0994 * num\_camas + 0.0006 * gasto\_salud + 77$$

$$MBA_{(h=0.1)} = 0.714$$

Caso 2. Hiperplano para $h = 0.6$:

$$\hat{y} = 0.45 * num\_medicos + 0.1496 * num\_camas + 0.0006 * gasto\_salud + 75.1$$

$$MBA_{(h=0.6)} = 0.789$$

En el modelo (SVR_3), se vuelve a repetir que el valor de $MBA < 1$. Cabe destacar que a mayor valor de h, se permite que la suma de los coeficientes ω sea mayor, implicando que el modelo sea menos preciso.

[8] rpubs.com/DashaIvanova/SVR3

Modelo		MBA Caso 1		MBA Caso 2
(SVR_1)	$C = 0.01$	0.938	$C = 0.1$	0.894
(SVR_2)	$c = 35$	0.57	$c = 40$	0.913
(SVR_3)	$h = 0.1$	0.714	$h = 0.6$	0.789

Tabla 3.6: Tabla resumen con los valores de la medida MBA para los diferentes modelos para los casos 1 y 2.

En la tabla 3.6 se pueden observar los siguientes detalles:

- Para el modelo (SVR_1), a mayor valor del parámetro C, menos errores se cometen al predecir y el modelo es más preciso.
- Para (SVR_2) ocurre que a mayor valor de c, permito que la suma de los errores sea mayor, y como consecuencia se darían peores predicciones al usar el modelo. Se puede apreciar en los valores de MBA para el caso 1 y 2.
- Para (SVR_3), a mayor valor de h, mayor valor para la medida MBA. Es decir, el modelo es menos preciso cuanto mayor sea el valor del parámetro h.
- Los valores de los MBA para los tres modelos y sus respectivos casos son relativamente similares, es decir, no hay valores que destaquen sobre los demás por ser demasiado grandes o demasiado pequeños.

Biblioteca
UNIVERSITAS Miguel Hernández

Conclusiones

Para concluir el TFG se aporta una valoración teórica y práctica de los resultados obtenidos en la aplicación sobre la esperanza de vida.

Se han podido realizar planteamientos alternativos al modelo cuadrático original de Support Vector Regression enfocándolo desde una perspectiva como un problema de optimización multiobjetivo. Usando algunas técnicas de resolución para PMO, se ha transformado el problema biobjetivo SVR a un problema de optimización con un solo objetivo, lo cual implica una simplificación a la hora de resolverlos.

La técnica de aprendizaje automático SVR expuesta en el trabajo resulta útil para realizar predicciones, además de ofrecer un margen de maniobra para jugar con los parámetros que intervienen en el modelo.

En cuanto a los resultados prácticos obtenidos en el capítulo III mediante los tres modelos estudiados a lo largo de los capítulos I y II, se concluye que:

- Mayor número de camas en hospitales, mayor número de médicos o mayor gasto en salud implica mayor esperanza de vida media.
- Cabe la posibilidad de obtener varias soluciones igual de válidas, en función de nuestros criterios a la hora de elegir los parámetros.
- No se aprecian grandes diferencias en los valores para la medida MBA en los diferentes modelos. Prácticamente todos se encuentran por debajo de 1, y no hay ningún valor excesivamente elevado.

De cara al futuro, sería interesante aplicar los modelos programados a un conjunto de datos con mayor número de registros. De esta forma, se podría implementar en la programación de los modelos planteados la técnica de *validación cruzada* mencionada en la introducción del trabajo, y obtener modelos que se ajusten mejor a nuevos datos.

Bibliografía

[1] Datos e información. https://rafaramoneblog.wordpress.com/2017/01/27/datos-e-informacion/. 2017.

[2] Las consecuencias del aumento de la esperanza de vida. https://noticias.universia.edu.uy/en-portada/noticia/2010/01/20/155603/consecuencias-aumento-esperanza-vida.html. 2010.

[3] Tipos de aprendizaje en machine learning: supervisado y no supervisado. https://empresas.blogthinkbig.com/que-algoritmo-elegir-en-ml-aprendizaje/. 2017.

[4] P. Barahona-Urbina. Factores determinantes de la esperanza de vida en chile. *Facultad de Ingeniería, Departamento de Industria y Negocios, Universidad de Atacama, Chile*, 2011.

[5] J. Bi. Multi-objective programming in svms. *Department of Mathematical Sicences, Rensselaer Polytechnic Institute, Troy, NY 12180 USA*, Washington DC, 2003.

[6] Y. G. García. Algoritmos svm para problemas sobre big data. *Universidad Autónoma de Madrid*, 2013.

[7] L. Laura-Ochoa. Evaluation of classification algorithms using cross validation. *Universidad Nacional de San Agustín de Arequipa, Perú*, 2019.

[8] E. C. León. Introducción a las máquinas de vector soporte (svm) en aprendizaje supervisado. *Universidad de Zaragoza*.

[9] D. Meyer. Support vector machines * the interface to libsvm in package e1071. *FH Technikum Wien, Austria*, 2019.

[10] E. J. C. Suárez. Tutorial sobre máquinas de vectores soporte (svm). *Dpto. de Inteligencia Artificial, ETS de Ingeniería Informática, Universidad Nacional de Educación a Distancia (UNED)*, 2016.

[11] H. Yu and S. Kim. Svm tutorial: Classification, regression, and ranking.

Lista de acrónimos

MOB	Modelo de Optimización Biobjetivo.
OCDE	Organización para la Cooperación y el Desarrollo Económicos.
PL	Programación Lineal.
PMO	Programación Multiobjetivo.
SVM	Support Vector Machine.
SVR	Support Vector Regression.
TFG	Trabajo Final de Grado.

Biblioteca
UNIVERSITAS Miguel Hernández

A. Anexo I: Código Modelo (SVR1)

Biblioteca
UNIVERSITAS Miguel Hernández

Support Vector Regression: Modelo (SVR1)

Lectura de datos:

```
datos = xlsread('data.xlsx');
```

Defino las variables en entrenamiento y testeo, y el parámetro epsilon:

```
[f,c] = size(datos);

P = 0.8;
idx = randperm(f);
train = datos(idx(1:round(P*f)),:);
test = datos(idx(round(P*f)+1:end),:);

x = train(:,1:(c-1));
y = train(:,c);

[n,d] = size(x)
```

```
n =
    25
d =
     3
```

```
x_test = test(:,1:d);
y_test = test(:,d+1);

epsilon = 0.5;
```

Biblioteca
UNIVERSITAS Miguel Hernández

Construcción del modelo (SVR1)

Función objetivo

Matriz variables no lineales:

```
H = zeros(d+1+2*n);
for i = 1:d
    H(i,i) = 1;
end
H;
```

Vector variables lineales:

```
% Este vector va a ir variando en función de los valores de C
% f = [zeros(1,d+1),repmat(C,1,2*n)]
```

Restricciones

(1) y-(<w,x>+b) <= epsilon+chi

```
A1 = [-x,-ones(n,1),-eye(n),zeros(n)];
```

(2) (<w,x>+b)-y <= epsilon + chi*

```
A2 = [x,ones(n,1),zeros(n),-eye(n)];
```

Matriz de coeficientes. Unión de las matrices A1 y A2:

```
A = [A1;A2];
```

Vector términos independientes:

```
b = [epsilon-y;epsilon+y];
```

Cotas no negatividad:

```
lb = [repmat(-Inf,1,d+1),zeros(1,2*n)];
```

Soluciones para diferentes valores de C

```
C = 0.01:0.01:10;
solution = zeros(length(C),1+d+1+2*n);
suma_w = 100000;

for i = 1:length(C)
    c = C(i);
    f = [zeros(1,d+1),repmat(c,1,2*n)];
    options = optimset('Display', 'off');
    sol = quadprog(H,f,A,b,[],[],lb,[],[],options);
    solt = transpose(sol);
    w = sum(solt(1,1:d).^2);
    if abs(w-suma_w)>0.000001 % Condición parada sum(w^2)[i]-sum(w^2)[i-1]<0.00001
        solution(i,1) = c;
        for j = 2:(d+1+2*n)
        solution(i,j) = solt(1,j-1);
        end
        suma_w = w;
    else
        break
    end
end

% Eliminamos aquellas filas de la matriz "solution" que sobran
ind = find(sum(solution,2)==0);
solution(ind,:) = [];

[f_sol,~] = size(solution)
```

```
f_sol =
    10
```

```
format shortG;
solution(1:3,1:(d+2)) % Muestro las primeras 3 soluciones
```

```
ans = 3×5
        0.01     0.051039     0.089212   0.00090323       75.817
        0.02     0.096819     0.099074   0.00087923       75.671
        0.03      0.14442      0.13644   0.00079698       75.569
```

```
solution((f_sol-3):f_sol,1:(d+2)) % Muestro las últimas 3 soluciones
```

```
ans = 4×5
        0.07      0.26939      0.15111   0.00072623       75.361
        0.08      0.29331      0.14034   0.00072458       75.336
        0.09      0.31722      0.12957   0.00072292       75.312
         0.1      0.32711      0.12512   0.00072223       75.301
```

Bondad de ajuste del modelo (SVR1)

Guardamos las variables calculadas:

```
C = solution(:,1);
w = solution(:,2:d+1);
b = solution(:,d+2);
```

Caso 1: hiperplano C más pequeño

Calculamos 'y' estimada para el valor de C más pequeño:

```
w_C_min = solution(1,2:(d+1));
b_min = solution(1,(d+2));
[n_test,~] = size(x_test);

y_est_1 = [];

for i = 1:n_test
    estimacion = w_C_min*transpose(x_test(i,:)) + b_min;
    y_est_1 = [y_est_1,estimacion];
end
y_est_1 = transpose(y_est_1)
```

```
y_est_1 = 6×1
       78.132
       78.619
       81.921
       80.632
       84.832
       77.499
```

Cálculo Medida Bondad Ajuste (MBA):

```
MBA_1 = ((mean(abs(y_test-y_est_1)))/(mean(abs(y_test-mean(y_test)))))
```

```
MBA_1 =
       0.9383
```

Caso 2: hiperplano C más grande

Calculamos 'y' estimada para el valor de C más grande:

```
[f_sol,c_sol] = size(solution);
w_C_max = solution(f_sol,2:(d+1));
b_max = solution(f_sol,(d+2));

y_est_2 = [];

for i = 1:n_test
    estimacion = w_C_max*transpose(x_test(i,:)) + b_max;
    y_est_2 = [y_est_2,estimacion];
end
y_est_2 = transpose(y_est_2)
```

```
y_est_2 = 6×1
       78.121
       78.796
       81.479
       80.332
       83.884
       78.233
```

Cálculo Medida Bondad Ajuste (MBA):

```
MBA_2 = ((mean(abs(y_test-y_est_2)))/(mean(abs(y_test-mean(y_test)))))
```

```
MBA_2 =
      0.89422
```

Biblioteca

B. Anexo II: Código Modelo (SVR2)

Biblioteca
UNIVERSITAS Miguel Hernández

Support Vector Regression: Modelo (SVR2)

Daria Ivanova

6/2020 - Trabajo Fin de Grado Estadística Empresarial

Librerías y lectura de datos

Cargamos las librerías que utilizaremos durante la construcción del modelo, especificando las funciones que usamos en cada una de ellas.

```
library(lpSolve) # lp
library(readxl) # read_xlsx
library(ggplot2) # ggplot
library(caret) # createDataPartition

# Para que no aparezcan los resultados en notación científica
options("scipen"=100, digits=3)
```

Leemos los datos y visualizamos parte de ellos.

```
data_paises <- read_xlsx('data.xlsx', col_names = TRUE)
head(data_paises)
```

```
## # A tibble: 6 x 5
##   paises_oecd num_medicos num_camas gasto_salud_per_cap esp_vida
##   <chr>             <dbl>     <dbl>               <dbl>    <dbl>
## 1 Alemania           3.92       8.2                4750.     80.5
## 2 Austria            4.84       7.7                4966.     80.9
## 3 Bélgica            2.94       6.3                4543.     80.4
## 4 Canadá             2.40       2.7                5353.     81.6
## 5 Colombia           1.74       1.5                 448.     75.9
## 6 Corea              2.08      10.3                1576.     80.8
```

Eliminamos la columna *paises_oecd* porque no nos aporta información relevante para el presente análisis.

```
data <- data.frame(subset(data_paises, select = -paises_oecd))
```

Definición de las variables y los parámetros

Definimos las variables conocidas, el parámetro ε y las dimensiones d y n. Además, se dividen los datos en entrenamiento y testeo.

```
# División datos entrenamiento/testeo
inTrain <- createDataPartition(y=data$esp_vida, p=0.8, list=FALSE)
train <- data[inTrain,]
test <- data[-inTrain,]
```

```r
# Variables conocidas
x <- train[, 1:ncol(train)-1]
y <- train[, ncol(train)]

d <- ncol(x) # Dimensión de las variables x
n <- nrow(x) # Tamaño de la muestra

epsilon <- 0.5 # Margen de error permitido
```

Construcción del problema de optimización (SVR2)

A continuación, vamos a definir el vector correspondiente a la función objetivo obj, y la matriz de coeficientes A y el vector de direcciones dir correspondientes a las restricciones.

Es importante tener en cuenta que el orden de las variables elegido es el siguiente:

$$x = (w_1^+ \quad \dots \quad w_d^+ \quad w_1^- \quad \dots \quad w_d^- \quad b^+ \quad b^- \quad \xi_1 \quad \dots \quad \xi_n \quad \xi_1^* \quad \dots \quad \xi_n^*)$$

Podemos ver que a la variable $b \in \mathbb{R}$ también le hemos aplicado la transformación del tipo $b := b^+ - b^-$, con $b^+, b^- \geq 0$. Esto es debido a que la función lp del paquete 'lpSolve' da por hecho que todas las variables son no negativas.

- Función objetivo: $\sum_{i=1}^{d}(\omega_i^+ + \omega_i^-)$

```r
obj <- c(rep(1,2*d), rep(0,2+2*n))
```

- Matriz de coeficientes:

Restricción (1): $\sum_{i=1}^{n}(\xi_i + \xi_i^*) \leq c$

```r
A1 <- c(rep(0,2*d+2),rep(1,2*n))
```

Restricción (2): $y_i - (\langle \omega^+ - \omega^-, x_i \rangle + (b^+ - b^-)) \leq \varepsilon + \xi_i, \quad i = 1,\dots,n$

```r
A2 <- cbind(-x,x,matrix(-1,n,1),matrix(1,n,1),-diag(n),matrix(0,n,n))
```

Restricción (3): $(\langle \omega^+ - \omega^-, x_i \rangle + (b^+ - b^-)) - y_i \leq \varepsilon + \xi_i, \quad i = 1,\dots,n$

```r
A3 <- cbind(x,-x,matrix(1,n,1),matrix(-1,n,1),matrix(0,n,n),-diag(n))
```

Unimos las restricciones $A_i, i = 1,2,3$ para formar la matriz final A:

```r
colnames(A2) <- 1:ncol(A2)
colnames(A3) <- 1:ncol(A2)

A <- rbind(A1,A2,A3)
```

- Vector de direcciones

```r
dir <- rep("<=", 1+2*n)
```

Nos falta definir el vector b, en el cual interviene el parámetro $c > 0$, que es la cota de la suma de los errores (ver Restricción (1)).

Para ello, se construye el siguiente bucle, donde vamos a ir dándole valores al parámetro c desde 1 hasta un valor grande y guardando las soluciones VÁLIDAS en la matriz $solutionC$, en función de dos condiciones:

(1) **Condición para que comience a almacenar las soluciones** en la matriz $solutionC$: el problema de optimización sea FACTIBLE. Para ello, se tiene que dar que el estado de la solución ($sol\$status$) tome el valor 0. Por lo tanto, el bucle comenzará dándole valores a $c = 1,2,3,\ldots$, para los cuales el estado de la solución sea no factible, hasta que llegue a un valor de c para el cual sí sea factible, y entonces se comenzará a guardar dicha solución.

(2) **Condición de parada**: |sol$objval - last$objval| < 0.000001. La diferencia, en valor absoluto, del valor de la función objetivo en la iteración (i) y el valor de la función objetivo en la iteración anterior (i-1) ha de ser lo suficientemente pequeña (0.000001) como para que ya no sea de interés seguir almacenando las soluciones para los diferentes valores de c. En el bucle lo definimos al revés, es decir, mientras la diferencia sea mayor que ese valor muy pequeño, que guarde la solución.

```
solutionC <- vector() # Vector vacío para almacenar w+, w- y b
chis <- vector() # Vector vacío para almacenar chi, chi*
last_objval <- .Machine$integer.max # Comparativa primera iteración, un
valor muy grande cualquiera (2147483647)

for (i in 1:last_objval){ # Valores para c
  b <- matrix(cbind(c(i, epsilon-y, epsilon+y)), ncol=1) # Vector
términos independientes b
  sol <- lp("min", obj, A, dir, b) # Resuelvo y guardo en 'sol' la
solución
  if (sol$status == 0){ # Primera condición
    if (abs(sol$objval-last_objval)>0.000001) # Segunda condición a la
inversa
    {
      solutionC <- rbind(solutionC, c(i, sol$solution[1:(2*d+2)])) #
Almaceno el valor de 'c' en la primera columna y la solución de w+,w- y b
      last_objval <- sol$objval # Guardo en 'last_objval' el valor de la
f. obj de la iteración actual
      chis <- rbind(chis,
c(sol$solution[(2*d+2+1):length(sol$solution)])) # Almaceno en 'chis' los
valores de chi,chi*
    } else {break} # Parada cuando deje de cumplirse la Segunda condición
a la inversa
  }
}

# head(solutionC) # Solución variables w+, w- y b
# head(chis) # Solución variables chi, chi*
```

De esta forma, ya tenemos las soluciones válidas para los diferentes valores de c almacenadas en el vector $solutionC$. Ahora, procedemos a deshacer las transformaciones realizadas anteriormente a las variables $\omega = \omega^+ - \omega^-$, $b = b^+ - b^-$, y guardar las soluciones definitivas de w, b en la matriz $solution\_fin$, y ξ, ξ^* en chi y chi_ast, respectivamente.

```
# Almaceno en solution_fin los valores de w, b con su correspondiente
valor c
solution_fin <- vector() # Vector vacío para w, b

for (i in 1:nrow(solutionC)){
  vector <- solutionC[i,1] # Columna valor c
  for (j in 2:(d+1)){
    vector <- cbind(vector, solutionC[i,j]-solutionC[i,j+d]) # Columnas
valores w+ - w-
  }
  vector <- cbind(vector, solutionC[i,ncol(solutionC)-1]-
solutionC[i,ncol(solutionC)]) # Columna valor b
  solution_fin <- rbind(solution_fin,vector)
}
solution_fin <- data.frame(matrix(solution_fin,ncol=1+d+1))
names(solution_fin) <- c("c","w1","w2","w3","b")
head(solution_fin)

##    c    w1     w2       w3    b
## 1 35 0.512 0.0216 0.000622 75.7
## 2 36 0.121 0.0246 0.000712 77.0
## 3 37 0.000 0.0000 0.000435 78.8
## 4 38 0.000 0.0000 0.000313 79.3
## 5 39 0.000 0.0000 0.000257 79.6
## 6 40 0.000 0.0000 0.000216 79.8

# Almaceno en chi y chi_ast sus correspondientes valores
chi <- chis[,1:(n)]
chi_ast <- chis[,(n+1):(ncol(chis))]
```

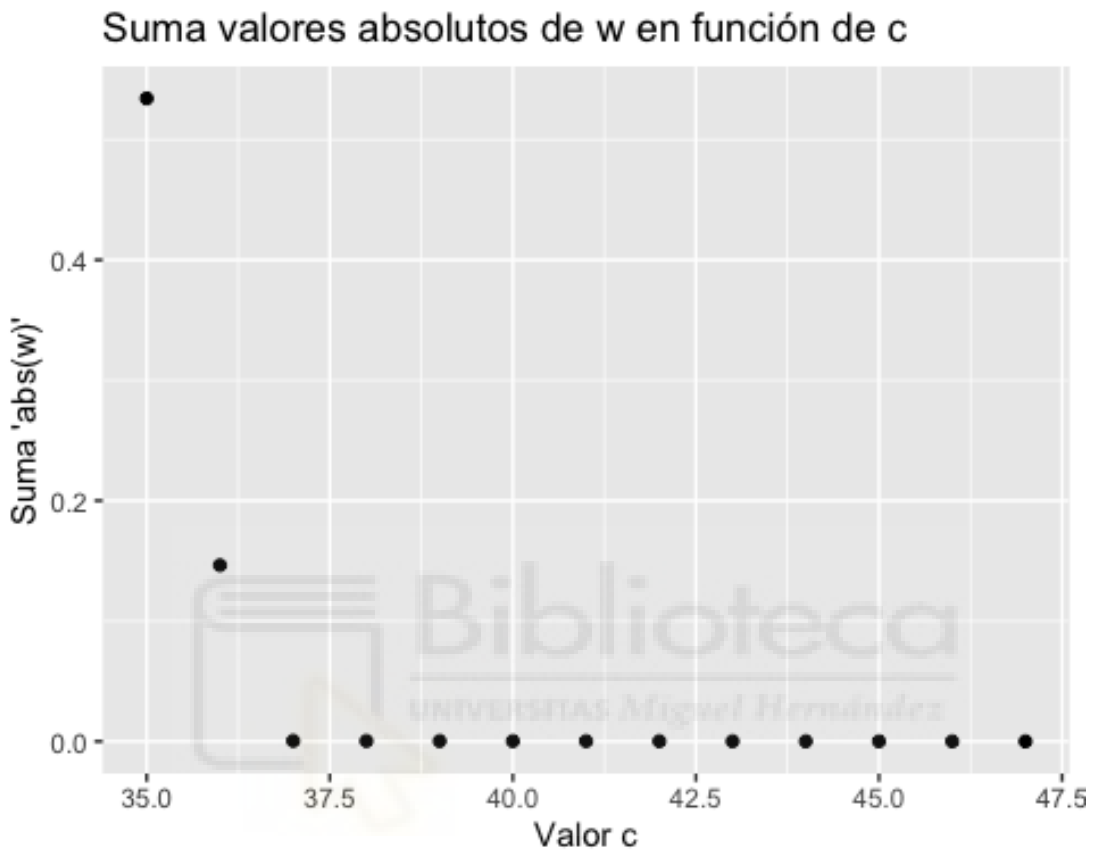
Gráfico $\sum_{i=1}^{d}|\omega_i|$ frente valores c

En este gráfico vamos a ver cómo evoluciona la suma de los valores de ω en valor absoluto, en función de los valores de c.

```
# Guardo en data.frames los correspondientes valores de la suma(w), b y
el parámetro c
w <- solution_fin[,2:(ncol(solution_fin)-1)]
sum_w <- data.frame(sum_w=rowSums(abs(w)))
b <- data.frame(b=solution_fin[,ncol(solution_fin)])
c <- data.frame(c=solution_fin[,1])
sum_w_c <- cbind(sum_w,c)

# Gráfico
```

```
ggplot(sum_w_c, aes(x=c, y=sum_w)) +
  geom_point() +
  labs(title="Suma valores absolutos de w en función de c", x="Valor c",
y="Suma 'abs(w)'")
```



A mayores valores de c, menor es la suma en valor absoluto de ω. En otras palabras, cuanto menos restrictivos somos con la restricción (1), más horizontal será el tubo que envuelve a los puntos.

Validación del modelo

Sobre los datos de entrenamiento

Para ver si el modelo es coherente con los datos que se han utilizado para "entrenarlo", vamos a comparar la diferencia entre real y estimado con los valores de ξ, ξ^* sumándoles ε. Es decir;

$$\text{Comparar } |\hat{y} - y| \text{ frente a } \xi + \varepsilon, \xi^* + \varepsilon$$

La comparativa la vamos a hacer para dos casos: la solución válida para el valor de c más pequeño y para el más grande.

Caso 1: hiperplano c más pequeño

Guardo la solución para el valor de c más pequeño.

```
w_C_min <- as.matrix(w[1,])
b_min <- as.matrix(b[1,])
chi_min <- as.matrix(chi[1,])
chi_ast_min <- as.matrix(chi_ast[1,])
chi_chi_ast <- cbind(chi_min,chi_ast_min)
sum_chis <- chi_min+chi_ast_min
```

Calculamos $\hat{y} = \langle \hat{\omega}, x_i \rangle + \hat{b}, \quad i = 1, \ldots, n$, para los datos de entrenamiento.

```
y_est_1 <- vector()
for (i in 1:nrow(x)){
  y_est_1 <- rbind(y_est_1, (w_C_min%*%t(x)[,i]) + b_min)
}
```

A continuación, calculamos la diferencia entre estimado y real, $|\hat{y} - y|$.

```
dif_est_real_1 <- abs(y_est_1-as.matrix(y))
```

Por último, hacemos la comparación $|\hat{y} - y|$ frente a $\xi + \varepsilon, \xi^* + \varepsilon$.

```
comparativa_1 <- data.frame(cbind(sum_chis+epsilon, dif_est_real_1),
row.names=NULL)
names(comparativa_1) <- c("Chi_Mas_Epsilon","Est_Menos_Real");
comparativa_1
```

```
##    Chi_Mas_Epsilon Est_Menos_Real
## 1            0.500         0.2979
## 2            0.500         0.5000
## 3            1.333         1.3334
## 4            1.020         1.0202
## 5            2.850         2.8499
## 6            1.799         1.7994
## 7            3.097         3.0967
## 8            3.553         3.5532
## 9            1.801         1.8006
## 10           0.500         0.4996
## 11           1.611         1.6112
## 12           0.500         0.5000
## 13           2.975         2.9751
## 14           0.500         0.4291
## 15           2.742         2.7418
## 16           2.544         2.5443
## 17           2.700         2.7002
## 18           4.098         4.0978
## 19           4.679         4.6789
## 20           2.086         2.0855
## 21           0.737         0.7371
## 22           1.320         1.3196
```

Biblioteca UNIVERSITAS Miguel Hernández

```
## 23           1.573         1.5725
## 24           0.500         0.5000
## 25           0.500         0.0227
## 26           0.880         0.8799
## 27           1.603         1.6028
```

```
dif_comparativa_1 <- as.data.frame(comparativa_1[,1]-comparativa_1[,2])
names(dif_comparativa_1) <- c("Diferencias"); round(dif_comparativa_1,
digits=3)
```

```
##    Diferencias
## 1        0.202
## 2        0.000
## 3        0.000
## 4        0.000
## 5        0.000
## 6        0.000
## 7        0.000
## 8        0.000
## 9        0.000
## 10       0.000
## 11       0.000
## 12       0.000
## 13       0.000
## 14       0.071
## 15       0.000
## 16       0.000
## 17       0.000
## 18       0.000
## 19       0.000
## 20       0.000
## 21       0.000
## 22       0.000
## 23       0.000
## 24       0.000
## 25       0.477
## 26       0.000
## 27       0.000
```

Biblioteca
UNIVERSITAS Miguel Hernández

Se puede observar que se cumplen las restricciones que se han definido en el modelo.

Caso 2: hiperplano *c* más grande

En este punto hacemos lo mismo que en el anterior, pero para el valor de *c* máximo dentro de nuestra matriz de soluciones.

```
w_C_max <- as.matrix(w[nrow(w),])
b_max <- as.matrix(b[nrow(b),])
chi_max <- as.matrix(chi[nrow(w),])
chi_ast_max <- as.matrix(chi_ast[nrow(w),])
```

```r
chi_chi_ast <- cbind(chi_max,chi_ast_max)
sum_chis <- chi_max+chi_ast_max

y_est_2 <- vector()
for (i in 1:nrow(x)){
  y_est_2 <- rbind(y_est_2, (w_C_max%*%t(x)[,i]) + b_max)
}

dif_est_real_2 <- abs(y_est_2-as.matrix(y))

comparativa_2 <- data.frame(cbind(sum_chis+epsilon, dif_est_real_2),
row.names=NULL)
names(comparativa_2) <- c("Chi_Epsilon","Est_Real"); comparativa_2
```

```
##    Chi_Epsilon Est_Real
## 1        0.500   0.1927
## 2        0.590   0.5902
## 3        1.302   1.3024
## 4        4.464   4.4643
## 5        0.500   0.4732
## 6        0.500   0.2220
## 7        2.080   2.0805
## 8        1.605   1.6049
## 9        4.020   4.0195
## 10       0.500   0.2805
## 11       1.622   1.6220
## 12       0.500   0.2878
## 13       5.283   5.2829
## 14       0.500   0.5000
## 15       1.359   1.3585
## 16       1.893   1.8927
## 17       2.750   2.7498
## 18       6.568   6.5683
## 19       6.483   6.4829
## 20       1.105   1.1049
## 21       3.600   3.6000
## 22       0.500   0.0268
## 23       0.559   0.5585
## 24       2.271   2.2707
## 25       1.359   1.3585
## 26       3.115   2.3512
## 27       4.973   4.9733
```

```r
dif_comparativa_2 <- as.data.frame(comparativa_2[,1]-comparativa_2[,2])
names(dif_comparativa_2) <- c("Diferencias");
round(dif_comparativa_2,digits=3)
```

```
##    Diferencias
## 1        0.307
## 2        0.000
## 3        0.000
```

```
## 4         0.000
## 5         0.027
## 6         0.278
## 7         0.000
## 8         0.000
## 9         0.000
## 10        0.220
## 11        0.000
## 12        0.212
## 13        0.000
## 14        0.000
## 15        0.000
## 16        0.000
## 17        0.000
## 18        0.000
## 19        0.000
## 20        0.000
## 21        0.000
## 22        0.473
## 23        0.000
## 24        0.000
## 25        0.000
## 26        0.764
## 27        0.000
```

De la misma forma, se cumplen las restricciones que se han definido en el modelo.

Sobre los datos de testeo

En esta última sección vamos a aplicar el modelo sobre los datos de testeo y a calcular una medida de bondad de ajuste para el mismo, definida en el TFG como MBA.

$$MBA = \left(\frac{1}{n}\sum_{i=1}^{n} |\, y_i - \hat{y}_i|\right) \frac{1}{\frac{1}{n}\sum_{i=1}^{n} |\, y_i - \overline{y}|}$$

Lo haremos, como en la sección anterior, para el caso de c más pequeño y para el más grande.

```
x_test <- test[,1:ncol(test)-1]
y_test <- test[,ncol(test)]
```

Caso 1: hiperplano c más pequeño

Calculamos $\hat{y} = \langle \hat{\omega}, x_i \rangle + \hat{b}, \quad i = 1, \ldots, n$, para los datos de testeo.

```
y_est_1 <- vector()
for (i in 1:nrow(x_test)){
  y_est_1 <- rbind(y_est_1, w_C_min%*%t(x_test)[,i] + b_min)
}
```

Y aplicamos la Medida de Bondad de Ajuste del modelo.

```
MBA_1 <- ((mean(abs(y_test-y_est_1)))/(mean(abs(y_test-mean(y_test)))));
MBA_1

## [1] 0.57
```

Caso 2: hiperplano *c* más grande

Realizamos los mismos pasos que en el punto anterior.

```
y_est_2 <- vector()
for (i in 1:nrow(x_test)){
  y_est_2 <- rbind(y_est_2, w_C_max%*%t(x_test)[,i] + b_max)
}

MBA_2 <- ((mean(abs(y_test-y_est_2)))/(mean(abs(y_test-mean(y_test)))));
MBA_2

## [1] 0.913
```

Biblioteca
UNIVERSITAS Miguel Hernández

C. Anexo III: Código Modelo (SVR3)

Biblioteca
UNIVERSITAS Miguel Hernández

Support Vector Regression: Modelo (SVR3)

Daria Ivanova

6/2020 - Trabajo Fin de Grado Estadística Empresarial

Librerías y lectura de datos

Cargamos las librerías que utilizaremos durante la construcción del modelo, especificando las funciones que usamos en cada una de ellas.

```
library(lpSolve) # lp
library(readxl) # read_xlsx
library(caret) # createDataPartition

# Para que no aparezcan los resultados en notación científica
options("scipen"=100, digits=3)
```

Leemos los datos y visualizamos parte de ellos.

```
data_paises <- read_xlsx('data.xlsx', col_names = TRUE)
head(data_paises)
```

```
## # A tibble: 6 x 5
##   paises_oecd num_medicos num_camas gasto_salud_per_cap esp_vida
##   <chr>             <dbl>     <dbl>               <dbl>    <dbl>
## 1 Alemania           3.92       8.2                4750.     80.5
## 2 Austria            4.84       7.7                4966.     80.9
## 3 Bélgica            2.94       6.3                4543.     80.4
## 4 Canadá             2.40       2.7                5353.     81.6
## 5 Colombia           1.74       1.5                 448.     75.9
## 6 Corea              2.08      10.3                1576.     80.8
```

Eliminamos la columna *paises_oecd* porque no nos aporta información relevante para el presente análisis.

```
data <- data.frame(subset(data_paises, select = -paises_oecd))
```

Definición de las variables y los parámetros

Definimos las variables conocidas, el parámetro ε y las dimensiones d y n. Además, vamos a dividir los datos en entrenamiento y testeo.

```
# División datos entrenamiento/testeo
inTrain <- createDataPartition(y=data$esp_vida, p=0.8, list=FALSE)
train <- data[inTrain,]
test <- data[-inTrain,]

# Variables conocidas
```

```r
x <- train[, 1:ncol(train)-1]
y <- train[, ncol(train)]

d <- ncol(x) # Dimensión de las variables x
n <- nrow(x) # Tamaño de la muestra

epsilon <- 0.5 # Margen de error permitido
```

Construcción del problema de optimización (SVR3)

A continuación, vamos a definir el vector correspondiente a la función objetivo obj, y la matriz de coeficientes A y el vector de direcciones dir correspondientes a las restricciones.

Es importante tener en cuenta que el orden de las variables elegido es el siguiente:

$$x = (w_1^+ \quad \dots \quad w_d^+ \quad w_1^- \quad \dots \quad w_d^- \quad b^+ \quad b^- \quad \xi_1 \quad \dots \quad \xi_n \quad \xi_1^* \quad \dots \quad \xi_n^*)$$

Podemos ver que a la variable $b \in \mathbb{R}$ también le hemos aplicado la transformación del tipo $b := b^+ - b^-$, con $b^+, b^- \geq 0$. Esto es debido a que la función lp del paquete 'lpSolve' da por hecho que todas las variables son no negativas.

- Función objetivo: $\sum_{i=1}^{n}(\xi_i + \xi_i^*)$

```r
obj <- c(rep(0,2*d+2),rep(1,2*n))
```

- Matriz de coeficientes:

Restricción (1): $\sum_{i=1}^{d}(\omega_i^+ + \omega_i^-) \leq h$

```r
A1 <- c(rep(1,2*d),rep(0,2+2*n))
```

Restricción (2): $y_i - (\langle \omega^+ - \omega^-, x_i \rangle + (b^+ - b^-)) \leq \varepsilon + \xi_i, \quad i = 1, \dots, n$

```r
A2 <- cbind(-x,x,matrix(-1,n,1),matrix(1,n,1),-diag(n),matrix(0,n,n))
```

Restricción (3): $(\langle \omega^+ - \omega^-, x_i \rangle + (b^+ - b^-)) - y_i \leq \varepsilon + \xi_i, \quad i = 1, \dots, n$

```r
A3 <- cbind(x,-x,matrix(1,n,1),matrix(-1,n,1),matrix(0,n,n),-diag(n))
```

Unimos las restricciones $A_i, i = 1,2,3$ para formar la matriz final A:

```r
colnames(A2) <- 1:ncol(A2)
colnames(A3) <- 1:ncol(A2)

A <- rbind(A1,A2,A3)
```

- Vector de direcciones

```r
dir <- rep("<=", 1+2*n)
```

Nos falta definir el vector b, en el cual interviene el parámetro $h > 0$, que es la cota de la suma de los errores (ver Restricción (1)).

Para ello, se construye el siguiente bucle, donde vamos a ir dándole valores al parámetro h desde 0.1 hasta un valor grande y guardando las soluciones VÁLIDAS en la matriz $solutionH$, en función de la siguiente condición:

Condición de parada: |sol$objval - last$objval| < 0.000001. La diferencia, en valor absoluto, del valor de la función objetivo en la iteración (i) y el valor de la función objetivo en la iteración anterior (i-1) ha de ser lo suficientemente pequeña (0.000001) como para que ya no sea de interés seguir almacenando las soluciones para los diferentes valores de c. En el bucle lo definimos al revés, es decir, mientras la diferencia sea mayor que ese valor muy pequeño, que guarde la solución.

```
solutionH <- vector() # Vector vacío para almacenar w+, w- y b
chis <- vector() # Vector vacío para almacenar chi, chi*
last_objval <- 10000 # Comparativa primera iteración frente a un valor
muy grande

for (i in seq(0.1,last_objval,0.1)){ # Valores para h
  b <- matrix(cbind(c(i, epsilon-y, epsilon+y)),ncol=1) # Vector términos
independientes b
  sol <- lp("min", obj, A, dir, b) # Resuelvo y guardo en 'sol' la
solución
  if (abs(sol$objval-last_objval)>0.000001) # Condición de parada a la
inversa
  {
    solutionH <- rbind(solutionH,c(i,sol$solution[1:(2*d+2)])) # Almaceno
el valor de 'h' en la primera columna y la solución de w+, w- y b
    last_objval <- sol$objval # Guardo en 'last_objval' el valor de la f.
obj de la iteración actual
    chis <- rbind(chis, c(sol$solution[(2*d+2+1):length(sol$solution)]))
# Almaceno en 'chis' los valores de chi,chi*
  } else {break} # Parada cuando deje de cumplirse la condición inversa
}

 head(solutionH) # Solución variables w+, w- y b
```

```
##      [,1]  [,2]   [,3]     [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]  0.1 0.000 0.0994 0.000622    0    0    0 77.0    0
## [2,]  0.2 0.000 0.1994 0.000644    0    0    0 76.3    0
## [3,]  0.3 0.127 0.1721 0.000646    0    0    0 76.1    0
## [4,]  0.4 0.231 0.1679 0.000660    0    0    0 75.7    0
## [5,]  0.5 0.327 0.1723 0.000656    0    0    0 75.3    0
## [6,]  0.6 0.450 0.1496 0.000622    0    0    0 75.1    0
```

```
# head(chis) # Solución variables chi, chi*
```

De esta forma, ya tenemos las soluciones válidas para los diferentes valores de h almacenadas en el vector $solutionH$. Ahora, procedemos a deshacer las transformaciones realizadas anteriormente a las variables $\omega = \omega^+ - \omega^-$, $b = b^+ - b^-$, y guardar las soluciones definitivas de w, b en la matriz $solution\_fin$, y ξ, ξ^* en chi y chi_ast, respectivamente.

```r
# Almaceno en solution_fin los valores de w, b con su correspondiente
valor h
solution_fin <- vector() # Vector vacío para w, b

for (i in 1:nrow(solutionH)){
  vector <- solutionH[i,1] # Columna valor h
  for (j in 2:(d+1)){
    vector <- cbind(vector, solutionH[i,j]-solutionH[i,j+d]) # Columnas
valores w+ - w-
  }
  vector <- cbind(vector, solutionH[i,ncol(solutionH)-1]-
solutionH[i,ncol(solutionH)]) # Columna valor b
  solution_fin <- rbind(solution_fin,vector)
}
solution_fin <- data.frame(matrix(solution_fin,ncol=1+d+1))
names(solution_fin) <- c("h","w1","w2","w3","b")
head(solution_fin)
```

```
##     h    w1     w2       w3    b
## 1 0.1 0.000 0.0994 0.000622 77.0
## 2 0.2 0.000 0.1994 0.000644 76.3
## 3 0.3 0.127 0.1721 0.000646 76.1
## 4 0.4 0.231 0.1679 0.000660 75.7
## 5 0.5 0.327 0.1723 0.000656 75.3
## 6 0.6 0.450 0.1496 0.000622 75.1
```

```r
# Almaceno en chi y chi_ast sus correspondientes valores
chi <- chis[,1:(n)]
chi_ast <- chis[,(n+1):(ncol(chis))]

# Guardo en diferentes vectores las variables calculadas y el parámetro h
w <- solution_fin[,2:(ncol(solution_fin)-1)]
b <- data.frame(b=solution_fin[,ncol(solution_fin)])
h <- data.frame(c=solution_fin[,1])
```

Validación del modelo

Sobre los datos de entrenamiento

Para ver si el modelo es coherente con los datos que se han utilizado para "entrenarlo", vamos a comparar la diferencia entre real y estimado con los valores de ξ, ξ^* sumándoles ε. Es decir;

$$\text{Comparar } |\hat{y} - y| \text{ frente a } \xi + \varepsilon, \xi^* + \varepsilon$$

La comparativa la vamos a hacer para dos casos: la solución válida para el valor de h más pequeño y para el más grande.

Caso 1: hiperplano h más pequeño

Guardo la solución para el valor de h más pequeño.

```
w_C_min <- as.matrix(w[1,])
b_min <- as.matrix(b[1,])
chi_min <- as.matrix(chi[1,])
chi_ast_min <- as.matrix(chi_ast[1,])
chi_chi_ast <- cbind(chi_min,chi_ast_min)
sum_chis <- chi_min+chi_ast_min
```

Calculamos $\hat{y} = \langle \hat{\omega}, x_i \rangle + \hat{b}, \quad i = 1, \ldots, n,$ para los datos de entrenamiento.

```
y_est_1 <- vector()
for (i in 1:nrow(x)){
  y_est_1 <- rbind(y_est_1, (w_C_min%*%t(x)[,i]) + b_min)
}
```

A continuación, calculamos la diferencia entre estimado y real, $|\hat{y} - y|$.

```
dif_est_real_1 <- abs(y_est_1-as.matrix(y))
```

Por último, hacemos la comparación $|\hat{y} - y|$ frente a $\xi + \varepsilon, \xi^* + \varepsilon$.

```
comparativa_1 <- data.frame(cbind(sum_chis+epsilon, dif_est_real_1),
row.names=NULL)
names(comparativa_1) <- c("Chi_Mas_Epsilon","Est_Menos_Real");
comparativa_1
```

```
##    Chi_Mas_Epsilon Est_Menos_Real
## 1            0.500         0.2758
## 2            0.500         0.0373
## 3            1.006         1.0062
## 4            1.593         1.5931
## 5            1.769         1.7691
## 6            1.402         1.4022
## 7            3.472         3.4716
## 8            3.839         3.8387
## 9            1.897         1.8970
## 10           0.500         0.3141
## 11           1.385         1.3846
## 12           1.926         1.9260
## 13           3.195         3.1949
## 14           2.866         2.8659
## 15           1.478         1.4776
## 16           4.319         4.3191
## 17           4.477         4.4769
## 18           0.582         0.5822
## 19           2.580         2.5804
## 20           1.569         1.5687
## 21           1.454         1.4537
## 22           1.795         1.7954
## 23           1.406         1.4062
## 24           0.500         0.5000
## 25           0.500         0.5000
```

```
## 26           0.599         0.5986
## 27           2.269         2.2690
```

```
dif_comparativa_1 <- as.data.frame(comparativa_1[,1]-comparativa_1[,2])
names(dif_comparativa_1) <- c("Diferencias"); round(dif_comparativa_1,
digits=3)
```

```
##    Diferencias
## 1        0.224
## 2        0.463
## 3        0.000
## 4        0.000
## 5        0.000
## 6        0.000
## 7        0.000
## 8        0.000
## 9        0.000
## 10       0.186
## 11       0.000
## 12       0.000
## 13       0.000
## 14       0.000
## 15       0.000
## 16       0.000
## 17       0.000
## 18       0.000
## 19       0.000
## 20       0.000
## 21       0.000
## 22       0.000
## 23       0.000
## 24       0.000
## 25       0.000
## 26       0.000
## 27       0.000
```

Biblioteca
UNIVERSITAS Miguel Hernández

Se puede observar que se cumplen las restricciones que se han definido en el modelo.

Caso 2: hiperplano h más grande

En este punto hacemos lo mismo que en el anterior, pero para el valor de h máximo dentro de nuestra matriz de soluciones.

```
w_C_max <- as.matrix(w[nrow(w),])
b_max <- as.matrix(b[nrow(b),])
chi_max <- as.matrix(chi[nrow(w),])
chi_ast_max <- as.matrix(chi_ast[nrow(w),])
chi_chi_ast <- cbind(chi_max,chi_ast_max)
sum_chis <- chi_max+chi_ast_max
```

```r
y_est_2 <- vector()
for (i in 1:nrow(x)){
  y_est_2 <- rbind(y_est_2, (w_C_max%*%t(x)[,i]) + b_max)
}

dif_est_real_2 <- abs(y_est_2-as.matrix(y))

comparativa_2 <- data.frame(cbind(sum_chis+epsilon, dif_est_real_2),
row.names=NULL)
names(comparativa_2) <- c("Chi_Epsilon","Est_Real"); comparativa_2
```

```
##    Chi_Epsilon Est_Real
## 1        0.500   0.5000
## 2        0.688   0.6881
## 3        1.880   1.8796
## 4        0.500   0.5000
## 5        2.378   2.3777
## 6        1.990   1.9897
## 7        3.413   3.4134
## 8        2.909   2.9090
## 9        1.808   1.8079
## 10       0.667   0.6674
## 11       1.644   1.6443
## 12       0.500   0.5000
## 13       3.349   3.3488
## 14       3.065   3.0655
## 15       2.008   2.0081
## 16       4.152   4.1521
## 17       4.961   4.9606
## 18       0.500   0.0509
## 19       1.659   1.6588
## 20       1.645   1.6453
## 21       0.808   0.8082
## 22       1.548   1.5476
## 23       2.012   2.0117
## 24       0.652   0.6519
## 25       0.500   0.4797
## 26       0.500   0.5000
## 27       1.219   1.2185
```

```r
dif_comparativa_2 <- as.data.frame(comparativa_2[,1]-comparativa_2[,2])
names(dif_comparativa_2) <- c("Diferencias");
round(dif_comparativa_2,digits=3)
```

```
##    Diferencias
## 1        0.000
## 2        0.000
## 3        0.000
## 4        0.000
## 5        0.000
## 6        0.000
```

Biblioteca UNIVERSITAS Miguel Hernández

```
## 7        0.000
## 8        0.000
## 9        0.000
## 10       0.000
## 11       0.000
## 12       0.000
## 13       0.000
## 14       0.000
## 15       0.000
## 16       0.000
## 17       0.000
## 18       0.449
## 19       0.000
## 20       0.000
## 21       0.000
## 22       0.000
## 23       0.000
## 24       0.000
## 25       0.020
## 26       0.000
## 27       0.000
```

De la misma forma, se cumplen las restricciones que se han definido en el modelo.

Sobre los datos de testeo

En esta última sección vamos a aplicar el modelo sobre los datos de testeo y a calcular una medida de bondad de ajuste para el mismo, definida en el TFG como MBA.

$$MBA = \left(\frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i|\right) \frac{1}{\frac{1}{n}\sum_{i=1}^{n} | y_i - \overline{y}|}$$

Lo haremos, como en la sección anterior, para el caso de h más pequeño y para el más grande.

```
x_test <- test[,1:ncol(test)-1]
y_test <- test[,ncol(test)]
```

Caso 1: hiperplano h más pequeño

Calculamos $\hat{y} = \langle \hat{\omega}, x_i \rangle + \hat{b}, \quad i = 1, \ldots, n$, para los datos de testeo.

```
y_est_1 <- vector()
for (i in 1:nrow(x_test)){
  y_est_1 <- rbind(y_est_1, w_C_min%*%t(x_test)[,i] + b_min)
}
```

Y aplicamos la Medida de Bondad de Ajuste del modelo.

```
MBA_1 <- ((mean(abs(y_test-y_est_1)))/(mean(abs(y_test-mean(y_test)))));
MBA_1
```

```
## [1] 0.714
```

Caso 2: hiperplano h más grande

Realizamos los mismos pasos que en el punto anterior.

```
y_est_2 <- vector()
for (i in 1:nrow(x_test)){
  y_est_2 <- rbind(y_est_2, w_C_max%*%t(x_test)[,i] + b_max)
}

MBA_2 <- ((mean(abs(y_test-y_est_2)))/(mean(abs(y_test-mean(y_test)))));
MBA_2

## [1] 0.789
```

Biblioteca
UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández