UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS

TESIS DOCTORAL

Optimización de la regresión de mínimos cuadrados parciales con funciones Kernel

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Jorge Daniel Mello Román

Director

Adolfo Hernández Estrada

Madrid

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS MATEMÁTICAS

TESIS DOCTORAL

OPTIMIZACIÓN DE LA REGRESIÓN DE MÍNIMOS CUADRADOS PARCIALES CON FUNCIONES KERNEL

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR:

JORGE DANIEL MELLO ROMÁN

DIRECTOR

DR. ADOLFO HERNÁNDEZ ESTRADA

PROGRAMA DE DOCTORADO EN INGENIERÍA MATEMÁTICA, ESTADÍSTICA E INVESTIGACIÓN OPERATIVA POR LA UNIVERSIDAD COMPLUTENSE DE MADRID Y LA UNIVERSIDAD POLITÉCNICA DE MADRID

OPTIMIZACIÓN DE LA REGRESIÓN DE MÍNIMOS CUADRADOS PARCIALES CON FUNCIONES KERNEL

TESIS DOCTORAL

JORGE DANIEL MELLO ROMÁN

Director

DR. ADOLFO HERNÁNDEZ ESTRADA

Madrid, 2020

A mi familia, padres, esposa, hijos y hermanos
su apoyo me mantuvo en el camino

A Dios,
somos instrumentos de su voluntad

Agradecimientos

Realizar el doctorado en la Universidad Complutense de Madrid, sin dudas ha marcado un hito en mi vida. Fueron cinco años enriquecedores y desafiantes, conciliando aprendizaje, trabajo y familia, he logrado fortalecer una identidad profesional y puedo vislumbrar mejor el horizonte de una carrera científica.

Llegar a escribir siquiera esta página, no hubiese sido posible sin la confianza del Profesor Adolfo, su paciencia, entusiasmo, motivación y orientaciones asertivas en todas las etapas del doctorado han sido claves para avanzar. Pero, sobre todo, por creer desde el primer día que era posible, a pesar de la distancia y sus implicaciones. ¡Gracias!

Hago una mención especial al Programa de Doctorado en Ingeniería Matemática, Estadística e Investigación Operativa y la Facultad de Ciencias Matemáticas de la UCM, por esa nobleza institucionalizada que se refleja en sus coordinadores, profesores y funcionarios. Sólo recibí aliento, buenos consejos, comprensión y ayuda en las dificultades.

Un enorme agradecimiento a toda mi familia y en especial a mi hermano Julio César, con quien además de los lazos de hermandad, comparto intereses científicos y mucho tiempo de colaboración mutua. Y a todas las demás personas e instituciones que de forma directa o indirecta me han brindado su apoyo en esta empresa, la suma de todo lo hizo posible.

Índice

Lista de Tablas

Lista de Figuras

Abreviaturas

PLS:	Partial Least Squares (EN)
	Mínimos cuadrados parciales (SP)
KPLS:	Kernel Partial Least Squares
	Mínimos cuadrados parciales con funciones kernel
GA:	Genetic Algorithms
	Algoritmos genéticos
PSO:	Particle Swarm Optimization.
	Optimización de enjambre de partículas.
FFA:	Firefly Algorithm
	Algoritmo de luciérnagas
GWO:	Grey Wolf Optimization.
	Algoritmo basado en el comportamiento de lobos grises
MMA:	Memetic Algorithms.
	Algoritmos Meméticos
HJ:	Hooke-Jeeves algorithm
	Algoritmo Hooke-Jeeves
NM:	Nelder-Mead algorithm
	Algoritmo Nelder-Mead

*(EN) Inglés, (SP) Español

Resumen

La regresión de mínimos cuadrados parciales (PLS) es un método lineal que busca predecir un conjunto de variables dependientes a partir de un conjunto de predictores, extrayendo factores ortogonales que maximizan la capacidad predictiva, también llamados componentes. Cuando las estructuras de datos exhiben variaciones no lineales, se recurre a la regresión de mínimos cuadrados parciales con kernel (KPLS), que transforma los conjuntos de datos originales a un espacio de características de dimensionalidad arbitraria donde sea posible la generación de un modelo lineal. Una dificultad recurrente al implementar la regresión KPLS es determinar el número de componentes y los parámetros de la función kernel que maximizan su desempeño.

El objetivo de esta investigación doctoral es proponer un método para optimizar la capacidad predictiva de la regresión KPLS por medio de algoritmos metaheurísticos. En el proceso de investigación se llevan a cabo procedimientos de ajuste metaheurístico para estimar simultáneamente los parámetros de la función kernel y el número de componentes. Se pone énfasis en la evaluación de algoritmos metaheurísticos debido a que los mismos obtienen mejores subconjuntos de búsqueda para determinar soluciones óptimas o aproximadas en el espacio transformado; además no requieren de supuestos, son flexibles y fáciles de implementar computacionalmente. Varias investigaciones han optado por utilizar algoritmos metaheurísticos para mejorar el desempeño de la regresión PLS o KPLS, aunque con propósitos y enfoques diferentes a los de esta tesis doctoral.

El problema de optimización es planteado y se evalúa el rendimiento de varios agentes de optimización en términos de precisión, convergencia de las estimaciones y velocidad de cómputo, tales como: Algoritmos genéticos (GA), Optimización de enjambre de partículas (PSO), Algoritmo de luciérnagas (FFA), Algoritmo basado en el comportamiento de lobos grises

(GWO), y Algoritmos Meméticos (MMA). Para validar los resultados y tener una medida de la eficiencia de los algoritmos metaheurísticos en el problema de optimización establecido, se evalúa también el desempeño de algoritmos determinísticos de búsqueda directa o libres de derivadas: Hooke-Jeeves (HJ) y Nelder–Mead (NM). Se recurre a estos métodos debido a que utilizan únicamente valores de la función y no precisan de una expresión explícita de las derivadas de la misma.

Los resultados experimentales indican que los algoritmos metaheurísticos permiten la estimación de valores óptimos aproximados de los parámetros de regresión del KPLS con una baja dispersión. Sin embargo, el costo computacional es elevado en comparación con otros algoritmos determinísticos. En general se logran los objetivos específicos de la tesis doctoral, una formulación novedosa del problema de optimización, la evaluación experimental del desempeño de diversos algoritmos metaheurísticos como agentes de optimización, y la creación de un paquete en el software R para la calibración de modelos de regresión KPLS con fines predictivos.

A partir del contenido de esta tesis doctoral han sido publicados una acta de conferencia y un artículo en revista de alto impacto, ambos relacionados directamente con los objetivos de investigación.

- Mello-Román, J. D., & Hernandez, A. (2020). KPLS optimization approach using genetic algorithms. *Procedia Computer Science*, *170*, 1153-1160. doi: 10.1016/j.procs.2020.03.051
- Mello-Román, J. D., & Hernandez, A. (2020). KPLS Optimization with Nature-Inspired Metaheuristic Algorithms. *IEEE Access*, 8, 157482-157492. doi: 10.1109/ACCESS.2020.3019771.

Según el Journal Citation Report del año 2019, la revista IEEE Access ocupa el lugar 35 de 156 títulos (Q1) en la categoría “Computer Science and Information Systems” y recibe un Factor de Impacto de 3,745.

Abstract

Partial Least Squares (PLS) regression is a linear method that seeks to predict a set of dependent variables from a set of predictors by extracting orthogonal factors that maximize predictive ability, also called components. When data structures exhibit non-linear variations, Kernel Partial Least Squares (KPLS) regression is used, which transforms the original data sets into an arbitrarily dimensioned feature space where a linear model can be generated. A recurring difficulty in implementing KPLS regression is determining the number of components and the parameters of the kernel function that maximize its performance.

The objective of this doctoral research is to propose a method to optimize the predictive capacity of KPLS regression by means of metaheuristic algorithms. In the research process, metaheuristic adjustment procedures are carried out to simultaneously estimate the parameters of the kernel function and the number of components. Emphasis is placed on the evaluation of metaheuristic algorithms because they obtain better search subsets to determine optimal or approximate solutions in the feature space, they do not require assumptions, they are flexible and easy to implement computationally. Several researches have chosen to use metaheuristic algorithms to improve the performance of PLS or KPLS regression, although with different purposes and approaches than those of this doctoral thesis.

The optimization problem is posed and the performance of several optimization agents is evaluated in terms of accuracy, convergence of estimates, and computational speed, such as: Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Firefly Algorithm (FFA), Grey Wolf Optimization (GWO), and Memetic Algorithms (MMA). To validate the results and have a measure of the efficiency of the metaheuristic algorithms in the established optimization problem, the performance of direct or derivative-free deterministic search algorithms is also

evaluated: Hooke-Jeeves (HJ) and Nelder-Mead (NM). These methods are used because they use only values of the function and do not require an explicit expression of the function's derivatives.

The experimental results indicate that the metaheuristic algorithms allow the estimation of approximate optimal values of the KPLS regression parameters with a low dispersion. However, the computational cost is high compared to other deterministic algorithms. In general, the specific objectives of the doctoral thesis, a novel formulation of the optimization problem, the experimental evaluation of the performance of several metaheuristic algorithms as optimization agents, and the creation of a package in the R software for the calibration of KPLS regression models for predictive purposes are achieved.

From the content of this doctoral thesis, a conference proceeding and an article in a high impact journal have been published, both directly related to the research objectives.

- Mello-Román, J. D., & Hernandez, A. (2020). KPLS optimization approach using genetic algorithms. *Procedia Computer Science*, *170*, 1153-1160. doi: 10.1016/j.procs.2020.03.051
- Mello-Román, J. D., & Hernandez, A. (2020). KPLS Optimization with Nature-Inspired Metaheuristic Algorithms. IEEE Access, 8, 157482-157492. doi: 10.1109/ACCESS.2020.3019771.

IEEE Access journal ranks 35th out of 156 titles (Q1) in the category "Computer Science and Information Systems" and receives an Impact Factor of 3.745, according to the Journal Citation Report of 2019.

Capítulo 1

Introducción

1.1. Estado del arte

El método de regresión de mínimos cuadrados parciales (PLS "Partial least squares") tiene su origen en el campo de la economía con los trabajos de Herman Wold (1975) en la década de los años sesenta. Actualmente goza de popularidad en las ciencias sociales, marketing y quimiometría (Pérez & González-Farias, 2013).

Pero la regresión de PLS es un método lineal y resulta inapropiada para describir estructuras de datos cuando estos exhiben variaciones con características no lineales (Gao et al., 2015). Para resolver el problema, Rosipal y Trejo (2001) proponen la regresión de mínimos cuadrados parciales con kernel (KPLS) que transforma los conjuntos de datos originales a un espacio de características de dimensionalidad arbitraria a través de un mapeo no lineal, y luego se crea un modelo lineal en el espacio de características (Zhang et al., 2017). La regresión KPLS se ha utilizado ampliamente en muchos campos científicos por su alta capacidad de generalización.

Shawe-Taylor y Cristianini (2004) manifiestan que la mejora de los métodos clásicos de estadística multivariante basados en la descomposición en valores singulares con el uso de funciones Kernel como KPLS, es un área de investigación muy recurrida en los últimos años. Asimismo, Bennet y Embrechts (2013) señalan que investigaciones futuras deben aprovechar al máximo el potencial de los algoritmos KPLS, que la teoría del aprendizaje estadístico puede

ayudar a clarificar sobre por qué KPLS generaliza bien, y que implementar variantes a KPLS podría resolver sus limitaciones.

Varios autores han combinado algoritmos metaheurísticos con modelos de regresión PLS o KPLS con diferentes propósitos y esquemas, Jalali-Heravi y Kyani (2007) y Zuvela et al (2016) combinaron algoritmos genéticos (Genetic Algorithms GA) y regresión KPLS, Wang et al. (2015) y Ying et al. (2017) utilizaron enjambre de partículas (Particle Swarm Optimization PSO) en modelos de regresión PLS, Goodarzi y dos Santos Coelho (2014), además de las metaheurísticas mencionadas, utilizaron algoritmo de luciérnagas (Firefly Algorithm FFA).

Sin embargo, un problema recurrente es determinar el número de componentes (variables latentes) y los parámetros de la regresión KPLS que maximizan la capacidad predictiva de los modelos generados (de Almeida et al., 2017). Huang et al. (2015) señalan que los principales problemas en la regresión KPLS son la selección de la función kernel y sus parámetros y la selección del número de componentes; sin embargo, la selección de la función kernel sigue siendo un problema abierto. En el mismo sentido encontramos que Rivas y Cerrillo (2014) señalan como problemas abiertos la selección de la función kernel y sus parámetros en técnicas multivariadas como la correlación canónica. Wei et al. (2008) plantean una problemática similar en modelos híbridos de clasificación y regresión multivariante.

El reciente trabajo de Fu et al. (2017) presenta una metodología de estimación simultánea del parámetro de la función kernel y el número de componentes en modelos KPCA y KPLS. Las diferencias de esta investigación doctoral con respecto a la propuesta de Fu et al. (2017) son varias, desde la formulación de la tarea de optimización y la función objetivo elegida hasta el enfoque de validación cruzada basado en la búsqueda en cuadrículas que propone. Debido a las diferencias en las bases de datos evaluadas y los paquetes de software utilizados no ha sido posible comparar ambas propuestas en términos de precisión de las estimaciones y costos computacionales asociados.

1.2. Objetivos

El objetivo general de la tesis doctoral es establecer un método de optimización de la capacidad predictiva de la regresión KPLS.

Los objetivos específicos son:

I. Formular el problema de optimización de la capacidad predictiva de la regresión KPLS.

   El rendimiento predictivo global de la regresión del KPLS puede evaluarse mediante un coeficiente de validación cruzada acumulativo Q^2_{cum}. Un aspecto novedoso de la tesis doctoral es que toma este coeficiente como función objetivo del problema de optimización, donde los argumentos son los parámetros de la función kernel y el número de componentes. Investigaciones anteriores han formulado tareas de optimización diferentes y escogido con frecuencia la suma de cuadrados del error de predicción como función objetivo.

II. Evaluar el desempeño de algoritmos metaheurísticos como optimizadores de la regresión KPLS.

   En el marco de este objetivo se diseña e implementa un algoritmo de estimación simultánea de los parámetros de la función kernel y el número de componentes de la regresión KPLS que maximizan el coeficiente Q^2_{cum}. Una característica distintiva de la propuesta es que el algoritmo se encuentra incorporado a un optimizador iterativo general, universal para cualquier función kernel y de fácil implementación computacional. En esta tesis doctoral se alcanza a evaluar el desempeño como optimizadores a los siguientes algoritmos metaheurísticos: Algoritmos genéticos (GA), Optimización de enjambre de partículas (PSO), Algoritmo de luciérnagas (FFA), Algoritmo basado en el comportamiento de lobos grises (GWO) y Algoritmos Meméticos (MMA). Se realiza un riguroso análisis estadístico del desempeño de estos optimizadores en términos de precisión y convergencia de las estimaciones, así como la velocidad de cómputo. Para validar los resultados y tener una referencia de la eficiencia de los algoritmos metaheurísticos como optimizadores, también se contrasta el desempeño de dos algoritmos determinísticos libres de derivadas: Hooke-Jeeves (HJ) y Nelder-Mead (NM).

III. Generar herramientas computacionales para calibrar modelos de regresión KPLS con fines predictivos.

Durante todo el proceso de investigación se desarrollan herramientas computacionales para resolver el problema de optimizar la capacidad predictiva de la regresión KPLS. Se acoplan funciones y paquetes del software R que permiten realizar las configuraciones y ejecuciones previstas en la formulación del problema y el marco experimental. Se documentan y comparten los códigos con el propósito de automatizar en el software R, el proceso de selección de parámetros óptimos de la regresión KPLS

1.4. Estructura de la memoria

La memoria de la tesis doctoral se organiza de la siguiente manera:

En el Capítulo 1 se presenta una breve revisión del estado del arte en la optimización de la regresión KPLS y se enuncian los problemas abiertos que han motivado la elección del tema de investigación. Se describen los objetivos, general y específicos, así como su abordaje.

En el Capítulo 2 se presentan los fundamentos de la regresión PLS y su formulación algorítmica, se introduce el concepto de función kernel y su incorporación a técnicas basadas en la descomposición en valores singulares de la matriz de covarianzas como la regresión KPLS. Seguidamente, se describe la estructura matemática de indicadores de la capacidad predictiva de la regresión KPLS, y en particular del coeficiente de validación cruzada acumulativo Q^2_{cum}.

En el Capítulo 3 se introducen los algoritmos metaheurísticos y determinísticos utilizados extensivamente en las evaluaciones experimentales como optimizadores. Se presenta la formulación matemática de cinco algoritmos metaheurísticos: GA, PSO, FFA, GWO y MMA y dos algoritmos libres de derivadas: NM y HJ.

En el Capítulo 4 se define del problema de optimización en función del coeficiente Q^2_{cum}, y se describen los pasos del algoritmo de estimación simultánea de los parámetros de la función kernel y el número de componentes. Seguidamente se presentan el soporte computacional del

software R y las herramientas generadas. El capítulo también recoge el marco metodológico de las evaluaciones experimentales.

El Capítulo 5 presenta los principales resultados experimentales alcanzados en toda la etapa de investigación. Se describen los principales conjuntos de datos evaluados, y la comparación del desempeño de los algoritmos en términos de precisión y convergencia de las estimaciones, así como velocidad de cómputo. El análisis estadístico de los resultados se desarrolla ampliamente en este capítulo.

El Capítulo 6 recoge las conclusiones de la tesis doctoral, un resumen de las principales contribuciones y publicaciones. El documento concluye con un análisis prospectivo de investigaciones y trabajos futuros.

Capítulo 2

Regresión de mínimos cuadrados parciales con kernel

Dada su importancia a lo largo de esta propuesta de investigación, este capítulo comienza presentando los fundamentos de la regresión de mínimos cuadrados parciales (PLS - Partial Least Squares) y su formulación algorítmica.

2.1. Regresión de mínimos cuadrados parciales

La regresión PLS es una técnica que generaliza y combina, características de Análisis de Componentes Principales y Regresión Lineal Múltiple (Abdi, 2010). A continuación, describiremos paso a paso su formulación matemática (Geladi y Kowalski, 1986).

En la regresión lineal múltiple (MLR - Multiple Linear Regression), dada una matriz $\mathbf{X}$ de variables independientes, un vector columna $\mathbf{y}$ que representa la variable dependiente, $\mathbf{b}$ el vector columna de los coeficientes y $\mathbf{e}$ el vector residual. Para n muestras y m variables independientes la relación lineal entre las variables puede plantearse conforme a la siguiente ecuación:

$$\mathbf{y}_{n\times 1} = \mathbf{X}_{n\times m}\mathbf{b}_{m\times 1} + \mathbf{e}_{n\times 1} \tag{1}$$

El método más popular para hacer esto se llama "Método de mínimos cuadrados" y su solución es:

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{2}$$

La ecuación (2) da una idea sobre el problema más frecuente en MLR, puede que no exista la inversa de $\mathbf{X'X}$. Es posible extender MLR para para r variables dependientes donde obtenemos el caso general:

$$\mathbf{Y}_{n\times r} = \mathbf{X}_{n\times m}\mathbf{B}_{m\times r} + \mathbf{E}_{n\times r} \tag{3}$$

$$\mathbf{B} = (\mathbf{X'X})^{-1}\mathbf{X'Y}$$

Por otra parte, en el análisis de componentes principales (PCA – Principal Component Analysis) (Peña, 2002), el problema consiste en definir un espacio de dimensión p más reducido que represente adecuadamente los datos. Este espacio estará determinado por los vectores propios asociados a los p mayores autovalores de la matriz de varianzas y covarianzas $\mathbf{S} = \frac{1}{n-1}\widetilde{\mathbf{X}}'\widetilde{\mathbf{X}}$ donde $\widetilde{\mathbf{X}}$ constituye la matriz centrada de los datos originales. Los componentes principales se obtienen realizando las siguientes operaciones:

$$|\mathbf{S} - \lambda\mathbf{I}| = 0 \tag{4}$$

$$(\mathbf{S} - \lambda_i\mathbf{I})\mathbf{a}_i = \mathbf{0}$$

Uno de los criterios para seleccionar el número de componentes es establecer una cota para la proporción de la variación explicada $\frac{\sum_i^p \lambda_i}{m}$, sin embargo esta regla es arbitraria. Llamando $\mathbf{T}$ a la matriz cuyas columnas son los valores de los p componentes en los n individuos, las nuevas variables están relacionadas con las originales mediante:

$$\mathbf{T}_{n\times p} = \mathbf{X}_{n\times m}\mathbf{P}_{m\times p} \tag{5}$$

Donde $\mathbf{P'P} = \mathbf{I}$, es decir los componentes principales son el resultado de aplicar una transformación ortogonal $\mathbf{P}$ a las variables $\mathbf{X}$ para obtener unas nuevas variables $\mathbf{T}$ incorreladas entre sí. Si utilizamos los nuevas variables $\mathbf{T}$ de la ecuación (5) en la fórmula de regresión lineal múltiple de la ecuación (3) obtenemos:

$$\mathbf{Y}_{n\times r} = \mathbf{T}_{n\times p}\mathbf{B}_{p\times r} + \mathbf{E}_{n\times r} \tag{6}$$

$$\widehat{\mathbf{B}} = (\mathbf{T'T})^{-1}\mathbf{T'Y}$$

Esta combinación de PCA y MLR es denominada regresión de componentes principales (PCR – Principal Component Regression). Las variables **X** son sustituidas por las nuevas variables **T** con mejores propiedades; son ortogonales, abarcan el espacio multidimensional de **X** y la expresión $\mathbf{T}'\mathbf{T}$ siempre es invertible. Pero los componentes son elegidos para explicar **X**, lo que no garantiza que serán pertinentes para explicar **Y** .

En el trabajo de Geladi y Kowalski (1986) se expone la regresión PLS como una regresión entre los componentes de las matrices de datos $\mathbf{X}_{n\times m}$ e $\mathbf{Y}_{n\times r}$, donde n es el número de muestras, con m variables independientes y r variables dependientes. Para p componentes se establecen las relaciones externas

$$\mathbf{Y}_{n\times r} = \mathbf{U}_{n\times p}\mathbf{Q}'_{p\times r} + \mathbf{F}_{n\times r} \tag{7}$$

$$\mathbf{X}_{n\times m} = \mathbf{T}_{n\times p}\mathbf{P}'_{p\times m} + \mathbf{E}_{n\times m}$$

donde las matrices **U** y **T** contienen los vectores componentes de **Y** y **X** respectivamente, con sus respectivas cargas **Q** y **P** y las matrices residuales **F** y **E**. Se establece además la relación lineal $\hat{\mathbf{u}}_h = b_h\mathbf{t}_h$ interna entre los vectores de **U** y **T** por cada componente p_h

El trabajo desarrollado por Herman Wold propone resolver la regresión PLS por medio del algoritmo NIPALS "Nonlinear Iterative Partial Least Squares". La descripción del algoritmo original se encuentra en la publicación de Geladi y Kowalski (1986). Desde allí se han hecho varios ajustes al algoritmo NIPALS propuesto por Wold (Koch, 2013).

Se destaca la versión de Lewi (1995) del algoritmo NIPALS presentada en la Tabla 1, que introduce una modificación al algoritmo original normalizando los vectores componentes.

Tabla 1. Algoritmo NIPALS según Lewi (1995)

Dadas las matrices de datos centradas **X** e **Y**

1. Iniciar aleatoriamente **u** (una columna de **Y**)
2. $\mathbf{w} = \mathbf{X}'\mathbf{u}$
3. $\mathbf{t} = \mathbf{X}\mathbf{w}$, $\mathbf{t} \leftarrow \frac{\mathbf{t}}{\|\mathbf{t}\|}$
4. $\mathbf{q} = \mathbf{Y}'\mathbf{t}$
5. $\mathbf{u} = \mathbf{Y}\mathbf{q}$, $\mathbf{u} \leftarrow \frac{\mathbf{u}}{\|\mathbf{u}\|}$
6. Repetir los pasos 2 - 5 hasta la convergencia
7. $\mathbf{X} \leftarrow (\mathbf{I} - \mathbf{t}\mathbf{t}'\mathbf{X})$; $\mathbf{Y} \leftarrow (\mathbf{I} - \mathbf{t}\mathbf{t}'\mathbf{Y})$ donde **I** es una matriz identidad

PLS es un proceso iterativo, es decir después de la extracción de cada componente, el algoritmo comienza de nuevo con las matrices calculadas en el paso 7. La matriz de coeficientes queda definida por la expresión:

$$\mathbf{B} = \mathbf{X}'\mathbf{U}(\mathbf{T}'\mathbf{X}\mathbf{X}'\mathbf{U})^{-1}\mathbf{T}'\mathbf{Y} \tag{8}$$

Es esta propuesta de Lewi (1995) la que toman Rosipal y Trejo (2001) para extender la regresión PLS para casos no lineales introduciendo el concepto de función kernel.

En la siguiente sección se introduce el concepto de función kernel en el desarrollo de la regresión de mínimos cuadrados parciales con kernel (KPLS – Kernel Partial Least Squares)

2.2. Regresión de mínimos cuadrados parciales con kernel

La regresión PLS es esencialmente un método de regresión lineal, se realiza bajo el supuesto de que los datos varían linealmente. Cuando las variaciones son no lineales, los datos pueden ser mapeados en un espacio de mayor dimensión en el que varíen linealmente (Bennet y Embrechts, 2013).

Dada la transformación no lineal $\varphi: \mathbb{R}^m \rightarrow \mathcal{H}$ a un espacio de Hilbert (provisto de producto interno) que permite proyectar los vectores de entrada desde el espacio de coordenadas original x a un espacio transformado $\varphi(x) \in \mathcal{H}$, el producto interno entre dos vectores dados en

el espacio transformado $\varphi(\mathrm{u})' \cdot \varphi(\mathrm{v}) = K(\mathrm{u}, \mathrm{v})$ enunciado en términos de una función de similitud en el espacio original, es conocido como función kernel (Tan et al., 2006).

El Teorema de Mercer demuestra que las funciones kernel pueden ser siempre expresadas como el producto escalar entre dos vectores dados en algún espacio altamente dimensional (Tan et al., 2006). Algunas de las funciones kernel más conocidas son:

- Gaussiana: $K(\mathrm{u}, \mathrm{v}) = \mathrm{e}^{-\|\mathrm{u}-\mathrm{v}\|^2/(2\sigma^2)}$ (8)
- Polinomial $K(\mathrm{u}, \mathrm{v}) = (k\mathrm{u} \cdot \mathrm{v} + \delta)^p$
- Laplaciana $K(\mathrm{u}, \mathrm{v}) = \mathrm{e}^{-\|\mathrm{u}-\mathrm{v}\|/\sigma}$
- Hiperbólica: $K(\mathrm{u}, \mathrm{v}) = \tanh(k\mathrm{u} \cdot \mathrm{v} - \delta)$

La incorporación de funciones kernel en técnicas basadas en la descomposición de valores singulares de la matriz de covarianzas se desarrolló de forma progresiva con el análisis de componentes principales con kernel (KPCA - Kernel Principal Component Analysis), la regresión de componentes principales con kernel (KPCR - Kernel Principal Component Regression) para extenderse finalmente a la regresión de mínimos cuadrados parciales con kernel (KPLS - Kernel Partial Least Squares) (Shawe-Taylor y Cristianini, 2004).

Rivas y Cerrillo (2014) definen la técnica KPCA como el análisis de componentes principales sobre datos transformados $\Phi(\mathbf{X})$, donde $\Phi: \mathbb{R}^m \rightarrow F$ es una transformación no lineal a un espacio de características F (espacio de Hilbert). La extensión a KPCR es la sustitución de las variables predictoras originales por los componentes principales obtenidos con KPCA. Una exposición detallada de los ambos métodos puede encontrarse en los textos de Shawe-Taylor y Cristianini (2004) y Koch (2013).

PLS sólo recientemente se ha extendido a la regresión no lineal a través del uso de funciones kernel K, denominada en este caso como regresión de mínimos cuadrados parciales con kernel (KPLS). El método propone esencialmente construir una función de regresión lineal en el espacio transformado que se corresponda a una función no lineal en el espacio de entradas original (Jia et al., 2015).

2.2.1. Definición matemática de la regresión KPLS

Dada una matriz $\mathbf{X}_{n\times m}$ de variables independientes $\{\mathbf{x}_i\}_{i=1}^{n} \in \mathbb{R}^m$ y una matriz $\mathbf{Y}_{n\times r}$ de variables respuesta $\{\mathbf{y}_i\}_{i=1}^{n} \in \mathbb{R}^r$.

Supongamos una transformación no lineal de las variables de entrada $\{\mathbf{x}_i\}_{i=1}^{n}$ en un espacio de características *F* provisto de producto interno; es decir, $\varphi: \mathbf{x}_i \in \mathbb{R}^m \rightarrow \varphi(\mathbf{x}_i) \in F$.

Denotamos por Φ una matriz (*n* × *M*) cuya *i*-ésima fila es el vector $\varphi(\mathbf{x}_i)$. Dependiendo de la transformación no lineal Φ(.) el espacio de características *F* puede ser altamente dimensional.

La función kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)' \cdot \varphi(\mathbf{x}_j)$ calcula el producto interno en el espacio de características *F*. La matriz Kernel $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}'$ representa una matriz (*n* × *n*) de los productos escalares cruzados de los datos transformados $\{\varphi(\mathbf{x}_i)\}_{i=1}^{n}$, es decir el elemento de la *i*-ésima fila y la *j*-ésima columna es la matriz kernel **K** es $k(\mathbf{x}_i, \mathbf{x}_j)$.

El algoritmo de la regresión KPLS propuesto por Rosipal y Trejo (2001) es como sigue:

Tabla 2. Algoritmo KPLS según Rosipal y Trejo (2001)

1. Iniciar aleatoriamente **u** (cualquier columna de **Y**)
2. $\mathbf{t} = \mathbf{K}\mathbf{u}$, $\mathbf{t} \leftarrow \frac{\mathbf{t}}{\|\mathbf{t}\|}$
3. $\mathbf{q} = \mathbf{Y}'\mathbf{t}$
4. $\mathbf{u} = \mathbf{Y}\mathbf{q}$, $\mathbf{u} \leftarrow \frac{\mathbf{u}}{\|\mathbf{u}\|}$
5. Repetir los pasos 2 - 5 hasta la convergencia
6. Deflactar las matrices **K** e **Y**
   $\mathbf{K} \leftarrow (\mathbf{I} - \mathbf{t}\mathbf{t}')\mathbf{K}(\mathbf{I} - \mathbf{t}\mathbf{t}')$; $\mathbf{Y} \leftarrow (\mathbf{I} - \mathbf{t}\mathbf{t}'\mathbf{Y})$ donde **I** es una matriz identidad *n*- dimensional
7. Repetir los pasos 2 - 7 hasta la extracción de *h* vectores de **U** y **T**

Después de la extracción de los *h* componentes, con las matrices generadas $\mathbf{T}_{n\times h}$ y $\mathbf{U}_{n\times h}$ queda definida la matriz **B** de coeficientes de la regresión KPLS por la siguiente expresión:

$$\mathbf{B} = \mathbf{\Phi}'\mathbf{U}(\mathbf{T}'\mathbf{K}\mathbf{U})^{-1}\mathbf{T}'\mathbf{Y} \tag{9}$$

Las predicciones sobre el conjunto de datos de entrenamiento se obtienen a partir de la fórmula:

$$\widehat{\mathbf{Y}} = \mathbf{K}\mathbf{U}(\mathbf{T}'\mathbf{K}\mathbf{U})^{-1}\mathbf{T}'\mathbf{Y} = \mathbf{T}\mathbf{T}'\mathbf{Y} \tag{10}$$

Para las predicciones en conjuntos de datos de prueba se utiliza la expresión:

$$\widehat{\mathbf{Y}}_{\boldsymbol{t}} = \mathbf{K}_{\boldsymbol{t}}\mathbf{U}(\mathbf{T}'\mathbf{K}\mathbf{U})^{-1}\mathbf{T}'\mathbf{Y} \tag{11}$$

donde la matriz $\mathbf{K}_{\boldsymbol{t}}$ es la matriz $(n_t \times n)$ cuyos elementos son las funciones kernel $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ evaluadas sobre los valores del conjunto de prueba $\{\mathbf{x}_i\}_{i=n+1}^{n+n_t}$y el conjunto de entrenamiento $\{\mathbf{x}_j\}_{i=1}^{n}$(Gao et al., 2015).

Antes de aplicar KPLS es necesario centralizar los datos en el espacio de las características (Schölkopf, et al., 1998). Para ello pueden aplicarse los siguientes procedimientos:

$$\mathbf{K} = \left(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n{}'\right)\mathbf{K}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n{}'\right) \tag{12}$$

$$\mathbf{K}_t = \left(\mathbf{K}_t - \frac{1}{n}\mathbf{1}_{n_t}\mathbf{1}_n{}'\mathbf{K}\right)\left(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n{}'\right)$$

donde $\mathbf{I}$ es de nuevo una matriz de identidad *n*-dimensional y $\mathbf{1}_n$, $\mathbf{1}_{n_t}$ representan los vectores cuyos elementos son unos, con longitud *n* y n_t, respectivamente.

La regresión KPLS ha demostrado a lo largo de estos años un alto desempeño para fines predictivos; no obstante, la selección del kernel óptimo y sus parámetros es un problema abierto (Bennet & Embrechts, 2013; Rivas & Cerrillo, 2014). Esta investigación doctoral se propone contribuir a la solución del problema mediante la aplicación de un algoritmo de optimización basado en metaheurísticas (Hernández Torres et al, 2014). Para ello es necesario definir primeramente la función matemática a optimizar.

A continuación, se describe la estructura matemática de algunos indicadores de la capacidad predictiva de la regresión PLS y KPLS, y en particular el coeficiente Q^2_{cum}.

2.2.2. Indicadores de la capacidad predictiva en KPLS

En cualquier modelización empírica, es esencial determinar la correcta complejidad del modelo. En la regresión PLS/KPLS es necesario comprobar la significación predictiva de cada componente, y no sumar componentes que no sean significativos. Según Abdi (2010), cuando el propósito de la regresión es generar modelos capaces de predecir el valor de las variables dependientes en nuevas observaciones, cada componente solo puede considerarse relevante si mejora esa predicción.

El procedimiento de validación cruzada es una forma práctica y fiable de probar la significación predictiva y se ha convertido en el estándar en PLS (Wold et al., 2001).

El procedimiento de validación cruzada es una de las técnicas de re-muestreo más antiguas (Stone, 1974). Consiste en dividir el conjunto de datos en segmentos de tamaño igual a *k* y luego utiliza bloques *k* - 1 para ajustar el modelo y validarlo en el segmento restante. Esto se hace para todas las combinaciones posibles de *k* - 1 de los *k* segmentos (Bischl et al., 2012). En este trabajo, el procedimiento de validación cruzada se aplicó para dividir aleatoriamente el conjunto de datos en *k* = 10 segmentos de igual tamaño (Xue & Yan, 2017).

La forma de evaluar la capacidad predictiva de un modelo es mediante la comparación de los valores observados y las predicciones del modelo (Consonni et al., 2010). Varias investigaciones toman como indicador la capacidad predictiva en PLS/ KPLS la raíz del error de cuadrático medio de predicción (RMSE – Root Mean Square Error) (Fu et al., 2017) o funciones de la misma (Jalali-Heravi & Kyani, 2007).

$$RMSEP = \sqrt{\frac{PRESS}{n}} = \sqrt{\frac{\sum_1^n (\hat{\mathbf{y}}_i - \mathbf{y}_i)^2}{n}} \tag{13}$$

donde $PRESS$ (Predictive Residual Sum of Squares) es la suma de las diferencias al cuadrado entre las predicciones $\hat{\mathbf{y}}_i$ y los valores observados $\mathbf{y}_i$.

Sin embargo, *RMSE* tropieza con el inconveniente de que sus valores dependen de las escalas de mediciones de las variables dependientes, y en ciertos estudios comparativos no es posible utilizarlo. Por esta razón, la capacidad de predicción de los modelos se cuantifica a

menudo en términos del coeficiente obtenido a partir de procesos de validación cruzada, denominado por algunos autores como coeficiente correlación cuadrática predictiva:

$$Q^2 = 1 - \frac{\sum_1^n (\hat{\mathbf{y}}_i - \mathbf{y}_i)^2}{\sum_1^n (\mathbf{y}_i - \bar{\mathbf{y}}_i)^2} = 1 - \frac{PRESS}{RSS} \tag{14}$$

donde RSS (Residual Sum of Square) es la suma de las diferencias al cuadrado de los valores observados $\mathbf{y}_i$ con respecto a la media $\bar{\mathbf{y}}$**.**

En la regresión KPLS/ PLS, el coeficiente Q^2 se calcula para cada componente h extraída. Su cálculo se realiza por medio de la siguiente fórmula (Tenenhaus, 1998):

$$\mathrm{Q}_h^2 = 1 - \frac{\mathrm{PRESS}_h}{\mathrm{RSS}_{h-1}} \tag{15}$$

donde RSS_{h-1} se calcula usando componente $h-1$ y PRESS usando el componente h .

Según Thévenot (2016), si el interés de la investigación se centra en evaluar la capacidad predictiva global del modelo de regresión PLS para un conjunto de $h > 1$ componentes, un indicador adecuado es el coeficiente $\mathrm{Q}_{\mathrm{cum}}^2$, que evalúa la capacidad de generalización del modelo para un conjunto de componentes $\{2, .. h-1, h\}$. Se obtiene por medio de la expresión matemática:

$$\mathrm{Q}_{cum}^2 = 1 - \prod_1^h (1 - Q_h^2) \tag{16}$$

El coeficiente Q_{cum}^2 toma valores entre 0 y 1, y cuanto más alto sea su valor, mejor será el rendimiento predictivo (Eriksson, et al., 2005).

Como ya se ha señalado anteriormente, un factor fundamental en la regresión PLS y KPLS es definir el número de componentes óptimo. En este sentido, uno de los criterios es considerar un componente significativo si $\mathrm{Q}_h^2 \geq 0{,}0975$ o de forma equivalente, mantener el componente h si $\sqrt{\mathrm{PRESS}_h} \leq 0{,}95\sqrt{\mathrm{RSS}_{h-1}}$ (Binard, 2012).

Otro criterio es considerar la cantidad óptima de componentes de acuerdo a la variación de los coeficientes Q_h^2 y $\mathrm{Q}_{\mathrm{cum}}^2$. El coeficiente $\mathrm{Q}_{\mathrm{cum}}^2$ se incrementa a medida que se agregan

componentes hasta cierto valor máximo para luego disminuir, y donde el punto crítico proporciona una estimación del número óptimo de componentes.

En este trabajo se opta por el segundo enfoque: evaluar la variación del coeficiente Q^2_{cum} y determinar por óptimo el número de componentes para el cual Q^2_{cum} alcanza su valor máximo. Este criterio es equivalente a considerar un componente significativo si $\sqrt{PRESS_h} \leq \sqrt{RSS_{h-1}}$ o $Q^2_h \geq 0$ (Thévenot, 2016).

Capítulo 3

Algoritmos metaheurísticos

En este capítulo se presentan los algoritmos metaheurísticos que forman parte de la tesis doctoral. Se definen los algoritmos genéticos (GA), la optimización de enjambre de partículas (PSO), el algoritmo de luciérnagas (FFA), el algoritmo basado en el comportamiento de lobos grises (GWO) y los algoritmos meméticos (MMA). Se incluyen breves referencias sobre algoritmos determinísticos libre de derivadas: Nelder–Mead (NM) y Hooke–Jeeves (HJ), que también integran esta memoria.

Se atribuye a Alan Turing ser el primero en usar algoritmos heurísticos durante la Segunda Guerra Mundial; llamó a su búsqueda "Heuristic Search" y fue una técnica muy utilizada (Shieber, 2004). El término algoritmo "metaheurístico" es acuñado en 1986 por F. Glover refiriéndose a los procedimientos que enriquecen los algoritmos heurísticos de modo que estos no queden atrapados en óptimos locales (Muñoz, 2007).

En su definición original, Metaheurística es un método de solución que orquesta una interacción entre los procedimientos de mejora local y las estrategias de nivel superior para crear un proceso capaz de escapar del óptimo local y realizar una búsqueda robusta de un espacio de solución (Gendreau & Potvin, 2010). Los algoritmos metaheurísticos son apropiados para resolver problemas de formulación compleja porque no utilizan ninguna información específica del problema en la exploración del espacio de soluciones factibles (Osaba et al., 2018).

Yang (2010) describe a los algoritmos metaheurísticos como algoritmos que generan diversas soluciones para explorar el espacio de búsqueda global e intensifican la búsqueda local

en una región donde tiene información que se encuentra una buena solución, asegurando de esta forma se alcancen óptimos globales. Según Asghari y Navimipour (2018), los algoritmos metaheurísticos son capaces de lidiar con grandes y complicados problemas, además son particularmente útiles cuando los métodos tradicionales se atascan en los mínimos locales.

Hay diferentes maneras de clasificar los algoritmos metaheurísticos. Una revisión exhaustiva de las clasificaciones puede encontrarse en el texto de Talbi (2009). Esta propuesta de investigación explora y evalúa primordialmente el desempeño de determinados algoritmos metaheurísticos basados en población e inspirados en la naturaleza.

Los algoritmos metaheurísticos inspirados en la naturaleza son aquellos que han sido desarrollados imitando el comportamiento de agentes biológicos (Kvasov & Mukhametzhanov, 2018). Según García-Ródenas et al. (2020), estos algoritmos son poderosos y eficaces para abordar diversos tipos de problemas de optimización, no requieren de supuestos para funcionar bien, son flexibles y fáciles de implementar. Los más populares son los algoritmos evolutivos y los de inteligencia de enjambre, que han demostrado su potencial para resolver importantes problemas de toma de decisiones en materia de ingeniería (Piotrowski et al., 2017).

Los algoritmos metaheurísticos basados en población se caracterizan por aplicar procedimientos de generación y reemplazo a una familia de soluciones esparcidas por el espacio de búsqueda. La mayoría de los algoritmos metaheurísticos basados en población (aunque no todos) están inspirados en la naturaleza (Gogna & Tayal, 2013).

En la revisión de la literatura científica es posible hallar varias investigaciones que han optado por algoritmos metaheurísticos para mejorar el desempeño de la regresión PLS/KPLS, aunque con objetivos y métodos diferentes a los planteados en esta tesis doctoral. Los algoritmos metaheurísticos más utilizados con problemas de regresión PLS/KPLS son probablemente: GA (Liu et al., 2015; Jalali-Heravi & Kyani, 2007; Žuvela et al., 2016) y PSO (Wang et al., 2015; Ying et al., 2017), y con menor frecuencia FFA (Goodarzi & dos Santos Coelho, 2014).

Según Lin et al. (2008) los enfoques metaheurísticos obtienen mejores subconjuntos de búsqueda en el espacio transformado para determinar soluciones óptimas o aproximadas, y además resultan adecuados para la tarea de determinar parámetros óptimos en modelos de regresión PLS (Žuvela et al., 2016).

Las características enunciadas respaldan el estudio de los algoritmos metaheurísticos como optimizadores de la regresión KPLS. Seguidamente se conceptualizan los algoritmos metaheurísticos y determinísticos que hacen parte de esta investigación.

3.1. Algoritmos genéticos

Holland (1975) desarrolló los algoritmos genéticos (GA), una metaheurística de base poblacional ampliamente utilizada en la actualidad en problemas de búsqueda y optimización. Goldberg (1989) define los algoritmos genéticos (GA) como algoritmos de búsqueda basados en la mecánica de selección natural y la genética natural, que combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado.

GA trabaja sobre una población de individuos (cromosomas) donde cada uno de ellos representa una solución potencial para el problema a resolver (Roeva & Fidanova, 2018). En el algoritmo, un individuo (o cromosoma) se representa por una cadena de bits binarios, se crea una población inicial de individuos al azar, se evalúa la función objetivo para cada individuo, y en base de estos valores, se producen los individuos de la siguiente iteración mediante operaciones de selección, cruce y mutación (Jalali-Heravi & Kyani, 2007). La Figura 1 ilustra el proceso en forma de diagrama de flujo.

Para la implementación de GA es necesario definir previamente sus hiperparámetros, como las probabilidades de cruce p_c y mutación p_m (Noorizadeh et al., 2013), así como el número inicial de población p y de generaciones g (Juan Yi et al 2017). El criterio de finalización puede estar determinado por el número de generaciones g o la convergencia de los resultados (Fizsewel et al., 2012).

Se destaca que GA ha sido frecuentemente utilizado en problemas de búsqueda y optimización de parámetros en modelos empíricos (Fogel, 2006; Frost & Molt, 1998).

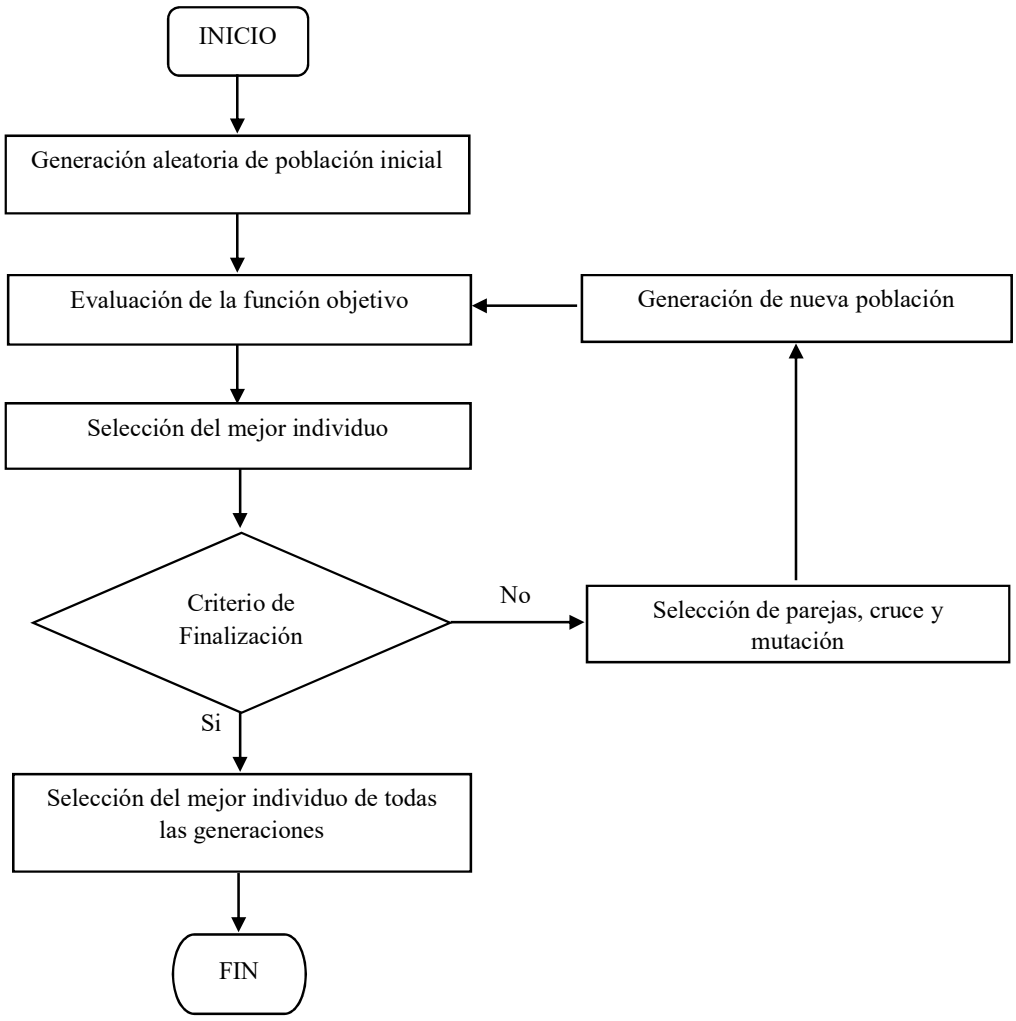


Figura 1. Diagrama de flujo. Optimización con algoritmos genéticos (GA)

3.2. Optimización de enjambre de partículas

Un progreso significativo en el campo de la metaheurística ha sido el desarrollo de la optimización de enjambre de partículas (PSO) por Eberhart y Kennedy (1995). PSO es una técnica de optimización metaheurística basada en población inspirada en el comportamiento social del vuelo de bandadas de aves o el movimiento de los bancos de peces. Desde su desarrollo se ha aplicado a casi todas las áreas de problemas de optimización difíciles.

En PSO cada partícula del enjambre es una solución potencial del problema de optimización, que se mueve por el espacio de búsqueda y actualiza su velocidad y posición con un mecanismo de aprendizaje basado en su mejor experiencia personal y en la mejor experiencia de la población (Wang et al., 2015). En cada iteración, cada partícula se mueve hacia una dirección que se basa en la mejor posición individual anterior $pbest$ y la mejor posición global $gbest$ del enjambre, para la dimensión d de la partícula i, la velocidad y la posición se pueden actualizar de acuerdo con las siguientes ecuaciones (Ying et al., 2017):

$$v_{id}^{t+1} = \omega \cdot v_{id}^{t} + c_1 \cdot r_1(pbest_{id}^{t} - x_{id}^{t}) + c_2 \cdot r_2(gbest_{id}^{t} - x_{id}^{t}) \quad (17)$$

$$x_{id}^{t+1} = v_{id}^{t+1} + x_{id}^{t}$$

donde t es la iteración, v_{id} la velocidad, x_{id} es la posición de la partícula, ω es el factor de inercia, c_1 y c_2 son factores de aprendizaje, y r_1 y r_2 son números aleatorios entre 0 y 1. El algoritmo PSO puede resumirse en los siguientes pasos:

(1) Iniciar el algoritmo para un número de población p, posición y velocidad de cada partícula.
(2) Calcular el valor de la función objetivo (o de aptitud) de la partícula i
(3) Comparar la función objetivo en la partícula y $pbest$. Si es mayor la función objetivo en la partícula, se reemplaza $pbest$
(4) Comparar la función objetivo en la partícula y $gbest$. Si es mayor la función objetivo en la partícula, se reemplaza $gbest$
(5) Actualizar la posición x_{id} y la velocidad v_{id}
(6) Repetir pasos 2 - 5 hasta un número de iteraciones g u otro criterio de convergencia.

3.3. Algoritmo de luciérnagas

El algoritmo de luciérnagas (FFA) es una técnica de inteligencia de enjambre, aplicada en la actualidad a diversas áreas de optimización y prácticas de ingeniería. Fue desarrollada por Yang (2010) y está inspirada en la imitación de las emisiones de luz utilizada por las luciérnagas para atraerse entre sí.

Para formular el algoritmo se establecen las siguientes condiciones: una luciérnaga puede ser atraída por cualquier otra luciérnaga, la atracción es proporcional al brillo para cualquier par de luciérnagas, la menos brillante se desplazará hacia la más brillante, a medida que las luciérnagas se alejan la percepción de la luz disminuye, y la intensidad de la luz emitida por la luciérnaga está determinada por el valor de la función a optimizar (Fister et al., 2013).

Cada luciérnaga (solución candidata) destella sus luces con cierto brillo (asociado a la función objetivo) atrayendo a otras luciérnagas dentro de su vecindario.

Este atractivo depende de la distancia entre las dos luciérnagas y está determinado por $\beta(r) = \beta_0 e^{-r_{ij}^2}$, donde r o r_{ij} es la distancia entre la *i*-ésima y la *j*-ésima luciérnaga, β_0 es el atractivo inicial $r = 0$ y γ es un coeficiente de absorción de luz fijo que controla la disminución de la intensidad de la luz (Kvasov & Mukhametzhanov, 2018).

El movimiento de la *i*-ésima luciérnaga atraída por otra luciérnaga más atractiva *j* viene dado por la siguiente ecuación:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j - x_i\right) + \alpha\epsilon_i \tag{18}$$

donde x_i es la ubicación particular de la *i*-ésima luciérnaga, α el parámetro de aleatorización, y ϵ_i es un vector de números aleatorios extraídos de una distribución gaussiana o uniforme.

Para una descripción más detallada de los pasos que sigue FFA puede recurrirse al texto de Vidal y Olivera (2018).

3.4. Algoritmo basado en el comportamiento de lobos grises

GWO es una técnica reciente de optimización metaheurística que simula el mecanismo jerárquico y el comportamiento depredador de la manada de lobos grises, donde bajo el liderazgo del lobo gris principal, los lobos capturan la presa a través de una serie de procesos, rodeando, cazando y atacando (Mirjalili et al., 2014). Se trata de un algoritmo con una gran capacidad de búsqueda global; sin embargo, se siguen realizando estudios para verificar su rendimiento en diferentes tipos de problemas y sugiriendo mejoras (Niu et al., 2019; Gao & Zhao, 2019).

Los lobos grises viven juntos y cazan en grupos, el proceso de búsqueda y caza puede describirse de la siguiente manera: primero los lobos grises rastrean a la presa, la persiguen y se acercan a ella, si la presa corre, entonces la persiguen, la rodean y la acosan hasta que deja de moverse. Finalmente, el ataque comienza (Mirjalili et al., 2014)

En el modelo matemático, la solución más apta, la segunda mejor y la tercera mejor se denominan: alfa (α), beta (β), y delta (δ) respectivamente. El resto de las soluciones candidatas se asumen todas como omegas (ω). Todos los ω son guiados por estos tres lobos grises durante la búsqueda (optimización) y la caza (Aquino, 2018).

Cuando se encuentra una presa comienza la iteración (t = 1). A partir de entonces, los lobos α, β y δ llevarían a los ω a perseguir y eventualmente rodear a la presa. Tres coeficientes $\vec{A}$, $\vec{C}$ y $\vec{D}$ describen el comportamiento circundante:

$$\vec{D}_\alpha = \left|\vec{C}_1.\vec{X}_a - \vec{X}(t)\right|; \quad \vec{D}_\beta = \left|\vec{C}_2.\vec{X}_\beta - \vec{X}(t)\right|; \quad \vec{D}_\delta = \left|\vec{C}_3.\vec{X}_\delta - \vec{X}(t)\right| \tag{19}$$

donde t indica la iteración actual, $\vec{X}$ es la posición vector del lobo gris, y $\vec{X}_1, \vec{X}_2$ y $\vec{X}_3$ son los vectores de posición de los lobos α, β y δ. La posición $\vec{X}$ se calcularía de la siguiente manera:

$$\vec{X}_1 = \vec{X}_a - \vec{A}_1.\vec{D}_\alpha; \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.\vec{D}_\beta; \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.\vec{D}_\delta \tag{20}$$

$$\vec{X}(t) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}$$

Los vectores $\vec{A}$ y $\vec{C}$ son combinaciones de un parámetro de control a y los números aleatorios $\vec{r}_1$ y $\vec{r}_2$.

$$\vec{A} = 2a.\vec{r}_1 - a; \quad \vec{C} = 2a.\vec{r}_2 \tag{20}$$

Una descripción extensa de la formulación original de GWO se halla en el texto de Mirjalili et al. (2014).

3.5. Algoritmos meméticos

Moscato (1989) introduce por primera vez el término algoritmos meméticos (MMA) para referirse a los algoritmos metaheurísticos que muestran un enfoque híbrido. MMA integra algoritmos evolutivos (como GA) y un procedimiento de búsqueda local para generar un algoritmo híbrido con características de ambos componentes. Este tipo de algoritmos es considerado un potente solucionador de problemas en el ámbito de la optimización continua, ya que ofrecen un equilibrio entre la exploración del espacio de búsqueda mediante su esquema de algoritmo evolutivo y la explotación enfocada de regiones prometedoras con un algoritmo de búsqueda local (Bergmeir et al., 2016).

Gogna y Tayal (2013) describen la metodología utilizada por MMA en los siguientes pasos genéricos: MMA comienza con la generación de una población inicial de soluciones (individuos). Esta población puede generarse de forma aleatoria o, para obtener mejores resultados, puede generarse utilizando un enfoque heurístico (específico del problema). El siguiente paso es la generación de una nueva población. Esto implica el uso de varias operaciones para generar descendientes. El primer operador es el operador de selección que selecciona el mejor individuo de la población de manera similar a la utilizada en los algoritmos evolutivos. Le sigue la recombinación de los individuos seleccionados para producir descendencia como en el caso de GA. Las crías generadas sufren mutaciones para aumentar la diversidad de la población. En el siguiente paso, se utiliza un operador de búsqueda local para causar variaciones aleatorias en la población generada. Esta operación se repite para buscar, en el entorno de cada individuo, una solución candidata con mejor aptitud que la solución original.

La Tabla 3 presenta un ejemplo de pseudocódigo para MMA. Para una revisión moderna y detallada se puede recurrir al texto de Moscato y Cotta (2010).

MMA está concebido en un paradigma ecléctico y pragmático, abierto a la integración de otras técnicas (metaheurísticas o no). Dada la amplitud de posibilidades, se destaca que el diseño algorítmico utilizado en esta investigación se basa en la propuesta en Bergmeir et al. (2016).

Tabla 3. Ejemplo de pseudocódigo para algoritmos meméticos (MMA)

```
ENTRADA: una instancia I de un problema P.
SALIDA: una solución sol.
// generar población inicial
para j ←1:popsize hacer
    sea ind ← GenerarSoluciónHeuristica (I)
    sea pop[j] ← MejoraLocal (ind, I)
finpara
repetir // bucle generacional
    // Selección
    sea criadores ← SeleccionarDePoblación (pop)
    // Reproducción segmentada
    sea auxpop[0] ← pop
    para j ← 1:#op hacer
        sea auxpop[j] ← AplicarOperador (op[j], auxpop[j - 1], I)
    finnpara
    sea newpop ← auxpop[#op]
    // Reemplazo
    sea pop ← ActualizarProblación (pop, newpop)
    // Comprobar convergencia
    si Convergencia (pop) entonces
        sea pop ← RefrescarPoblación (pop, I)
    finnsi
hasta CriterioTerminación (pop, I)
devolver Mejor (pop, I)
```

3.6. Algoritmos libres de derivadas

Si bien esta investigación se enfoca en la optimización de la regresión KPLS por medio de algoritmos metaheurísticos, es necesario, tanto para validar los resultados como tener una referencia de eficiencia, evaluar el desempeño de optimizadores con una naturaleza diferente, como los algoritmos determinísticos. Dada la complejidad y dimensión del problema de optimización planteado se recurre específicamente a los algoritmos libres de derivadas.

Los algoritmos libres de derivadas, también llamados de búsqueda directa, utilizan solo valores de función y se aplican cuando no se puede producir un código informático para la

derivada de la función. En general son métodos robustos, aún para funciones no convexas o con discontinuidades (Larson et al., 2019)

Según Sun et. al (2020), cuando es necesario determinar el óptimo de una función de valores reales definida en un espacio *n*-dimensional, pero las derivadas no están disponibles porque son computacionalmente costosas o no se tiene una expresión explícita de las derivadas parciales de la función, es posible recurrir a métodos de optimización llamados libres de derivadas o de búsqueda directa.

Dos algoritmos de optimización libres de derivadas muy populares: Nelder-Mead (NM) y Hooke-Jeeves (HJ) (Rios & Sahinidis, 2013) son incluidos en el marco experimental de esta investigación.

El algoritmo NM es un método de búsqueda directa para la optimización sin restricciones de funciones multidimensionales, y que trata de minimizar una función escalar no lineal de *n* variables usando sólo valores de la función, sin obtener ninguna información de la derivada (Nelder, 1965).

El método HJ es también un método de optimización libre de derivadas, que busca las direcciones de descenso mediante movimientos exploratorios a través de las direcciones coordenadas realizando dos tipos de búsqueda: una búsqueda exploratoria y una búsqueda de patrones, repitiendo hasta la convergencia (Hooke, 1961).

Capítulo 4

Problema de optimización

Los problemas principales en KPLS son: la selección de la función kernel y sus parámetros y la selección del número de componentes (Huang et al., 2015). Esta investigación propone un procedimiento de ajuste metaheurístico para la estimación simultánea de los parámetros de la función kernel y el número de componentes.

Un aspecto distintivo de la propuesta en comparación a otros trabajos (Fu et al., 2017; Huang et al., 2015) es el problema de optimización planteado, que toma como función objetivo el coeficiente $\mathrm{Q}^2_{\mathrm{cum}}$ y estudia el desempeño de varios algoritmos metaheurísticos como agentes de optimización.

4.1. Formulación del problema

Dadas las matrices de datos originales $\mathbf{X}_{n\times m}$ y $\mathbf{Y}_{n\times r}$, y una función kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ de parámetro θ generadora de la matriz kernel $\mathbf{K}_{n\times n}$.

Efectuada la regresión KPLS conforme a la Tabla 2 y extraídos h componentes, se establece como función objetivo el coeficiente $\mathrm{Q}^2_{\mathrm{cum}}(h,\theta)$ determinado por la Ecuación 16, el cual se halla función de los valores que toman h y θ en la regresión KPLS.

La tarea de optimización puede escribirse de la siguiente manera:

$$(h,\theta) = \underset{h,\theta}{\operatorname{argmax}}\, \mathrm{Q}^2_{\mathrm{cum}} \tag{21}$$

$$\theta \in S \subset \mathbb{R}$$

$$h \in \mathbb{Z}^+ \leq m$$

donde número de componentes h puede tomar valores enteros positivos menores al número de columnas de la matriz $\mathbf{X}$ y el dominio del parámetro de la función kernel θ es un subconjunto predefinido S de números reales.

La función objetivo Q^2_{cum} es un indicador de la capacidad predictiva global de la regresión KPLS, toma valores entre 0 y 1 y cuanto mayor su valor, mayor capacidad predictiva del modelo (Eriksson, et al., 2005). Se obtiene a partir de procedimientos de validación cruzada y es adimensional, es decir, no es afectado por las escalas de las variables de respuesta.

La formulación del problema de optimización de la capacidad predictiva de la regresión KPLS es el primer objetivo específico descrito en esta memoria.

4.1.1. Algoritmo de selección de parámetros

Una característica adicional de esta propuesta es que diseña un algoritmo ya incorporado en un optimizador iterativo general, por ejemplo, un algoritmo metaheurístico basado en población. El método propuesto es universal para cualquier función kernel y fácil de implementar computacionalmente. La Tabla 4 describe los pasos del método propuesto:

Tabla 4. Algoritmo de selección de parámetros en la regresión KPLS

Dadas las matrices de datos originales $\mathbf{X}_{n\times m}$ y $\mathbf{Y}_{n\times r}$, y una función kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ de parámetros $\theta \in S \subset \mathbb{R}$.

1. El optimizador genera aleatoriamente una población inicial p de θ (individuos)
2. A partir de la matriz $\mathbf{X}_{n\times m}$ se generan p matrices kernel $\mathbf{K}_{n\times n}$, una por cada θ
3. Se implementa la regresión KPLS (Tabla 2) entre $\mathbf{K}_{n\times n}$ y $\mathbf{Y}_{n\times r}$ con $h \in \mathbb{Z}^+ \leq m$ componentes.
4. Q^2_{cum} es calculado por medio de validación cruzada por cada KPLS.
5. Se identifican los θ asociados al mayor Q^2_{cum} y se extrae el número de componentes para el cual se alcanza este valor.
6. Se realizan g iteraciones, siendo la función objetivo $\max Q^2_{\text{cum}}$
7. El optimizador determina los mejores valores de θ y h y el valor óptimo de Q^2_{cum}

El algoritmo descrito en la Tabla 4 tiene un carácter genérico. Puede utilizarse con las funciones kernel gaussiana, polinomial, laplaciana e hiperbólica, descritas en la Sección 2.2. u otras que se encuentran en la literatura científica (Zeileis et al., 2004).

Si bien el algoritmo está concebido para optimizadores metaheurísticos basados en población, con ajustes es posible integrar algoritmos de búsqueda directa en un esquema similar. Para ello se establece un conjunto aleatorio de puntos de partida a fin de reducir el atascamiento en óptimos locales (Asghari y Navimipour, 2018).

En el Capítulo 5 se evalúa el rendimiento de varios optimizadores (metaheurísticos y determinísticos) integrados al algoritmo de selección de parámetros propuesto en la Tabla 4.

4.2. Marco experimental

El marco experimental se circunscribe al conjunto de buenas prácticas para la presentación y comparación de metaheurísticas, propuesto por Osaba et al. (2018). Las características más importantes se resumen a continuación:

1. El desempeño de los algoritmos metaheurísticos se evalúa tomando como referencia algoritmos no metaheurísticos.
2. Se incluye información sobre el número de ejecuciones realizadas, los mejores resultados, media y desviación estándar de las estimaciones y otros indicadores.
3. Se evalúa el comportamiento de la convergencia de cada técnica utilizada.
4. Las experimentaciones se realizan sobre varios conjuntos de datos.
5. Se muestran tiempos de ejecución y las características del ordenador en el que se han realizado las pruebas.
6. Se realiza un análisis estadístico con los resultados obtenidos.
7. Se comparte el código de los algoritmos implementados.

Las evaluaciones y comparaciones experimentales se desarrollan durante todo el proceso de investigación de manera independiente. Se organizan en tres grandes partes presentadas en el capítulo siguiente:

A. Optimización con algoritmos genéticos (GA):

En esta sección se proponen a GA como optimizador de la regresión KPLS. Algunas características diferenciadoras son:

a) Se agregan pruebas preliminares para configurar el tamaño de la población p y de número de generaciones g de GA.
b) Los hiperparámetros de GA se determinan con referencia a investigaciones relacionadas.
c) Se toma el máximo Q^2_{cum} de cada ejecución y se establece un criterio de convergencia en términos de la dispersión relativa en las estimaciones.
d) Se comparan con resultados obtenidos tanto por los algoritmos NM y HJ como PLS simple.

B. Optimización con algoritmos inspirados en la naturaleza**:**

En esta sección se compara el desempeño de los algoritmos GA, PSO, FFA, GWO como optimizadores de la regresión KPLS. Algunas características distintivas son:

a) Se agregan pruebas preliminares para configurar los hiperparámetros de las metaheurísticas.
b) Se obtienen resultados para un tamaño de población fijo p y cantidades incrementales de iteraciones g.
c) Se evalúan la media y la desviación estándar de las estimaciones.
d) La convergencia de las estimaciones se analiza con respecto al incremento de g
e) Se realiza un extenso análisis estadístico comparativo de los resultados.

C. Optimización con algoritmos meméticos (MMA):

En esta sección se evalúa el desempeño del algoritmo MMA como optimizador de la regresión KPLS. Algunas características distintivas son:

a) Utiliza los resultados de las evaluaciones experimentales anteriores.
b) Las comparaciones se realizan para valores constantes del tamaño de población p y cantidades de iteraciones g.
c) Se evalúan la media, la desviación estándar y el coeficiente de variación de las estimaciones.

d) El análisis estadístico se enfoca en la precisión de las estimaciones y el tiempo de ejecución del algoritmo.

Algunas características comunes del proceso experimental en las tres etapas mencionadas son las siguientes:

a) Se realizan k = 30 (treinta) ejecuciones del algoritmo presentado en la Tabla 4, a fin de evaluar la variación y convergencia de las estimaciones (Vidal & Olivera, 2017).
b) Se realizan pruebas preliminares para la configuración de las metaheurísticas a fin de evitar sesgos en los resultados.

En la revisión de la literatura científica se encuentra que la selección de hiperparámetros en algoritmos metaheurísticos se realiza: de forma arbitraria, tomando como referencia investigaciones similares o realizando pruebas preliminares (Attia et al., 2017; Yi et al., 2017)

De igual manera se encuentran estudios preliminares para determinar el número de población y el número máximo de iteraciones (Carrero Yubero, 2011). El tamaño de la población p debe ser adecuado tanto para garantizar la diversidad de soluciones como evitar la ralentización del algoritmo (Merelo Guervós, 2009; Arranz de la Peña & Parra Truyol, 2007). En cuanto al número de iteraciones g, es posible determinarlo arbitrariamente o conforme algún criterio de convergencia (Fiszelew et al.,2002)

4.2.1. Análisis estadístico

Para realizar los estudios comparativos es necesario obtener información sobre la tendencia central y la variación de las estimaciones. Las estadísticas descriptivas utilizadas son:

a) La media $\bar{\theta} = \frac{\sum_1^k \theta_i}{k}$, la desviación estándar $s = \sqrt{\frac{\sum_1^k (\theta_i - \bar{\theta})^2}{k-1}}$ de las estimaciones, donde k es el número de ejecuciones (Osaba et al., 2018; Kvasov & Mukhametzhanov, 2018).
b) El coeficiente de variación determinado por el cociente entre la desviación estándar y la media $CV = \frac{s}{\bar{\theta}}$.

c) El máximo θ^* de las estimaciones y una medida de la dispersión relativa con respecto a θ^*, propuesta en esta investigación:

$$D = \frac{1}{\theta^*}\sqrt{\frac{\sum_1^k(\theta_i-\theta^*)^2}{k}} \tag{22}$$

En el Capítulo 5 se presentan tablas y gráficos sobre las estadísticas descriptivas. La convergencia de las estimaciones de θ, h y $\mathrm{Q}_{\mathrm{cum}}^2$ es evaluada con respecto al incremento gradual del número de iteraciones y/o el tamaño de la población. Las pruebas estadísticas aplicadas sobre los resultados obtenidos se describen a continuación (Lee & Lee, 2018; Shingala, & Rajyaguru, 2015; Ramsey & Ramsey, 2008):

a) Análisis de varianza (ANOVA): contrasta la hipótesis de igualdad de medias en las estimaciones de θ, h y $\mathrm{Q}_{\mathrm{cum}}^2$ con respecto a uso de diferentes optimizadores. Se evalúa la misma hipótesis con respecto al incremento del número de iteraciones en los algoritmos metaheurísticos.
b) Prueba de Levene o de homogeneidad de varianzas: contrasta la hipótesis de igualdad de varianzas en las estimaciones de θ, h y $\mathrm{Q}_{\mathrm{cum}}^2$ con respecto a uso de diferentes optimizadores. Se evalúa la misma hipótesis con respecto al incremento del número de iteraciones en los algoritmos metaheurísticos.
c) Comparaciones múltiples post hoc: contrasta la hipótesis de igualdad de medias en las estimaciones $\mathrm{Q}_{\mathrm{cum}}^2$ por pares de optimizadores. Se utiliza la prueba T2 Tamhane indicada cuando las varianzas y/o los tamaños muestras son desiguales.
d) Prueba de Kolmogorov-Smirnov o de normalidad: contrasta la hipótesis de la distribución normal de las estimaciones de θ con respecto al uso de diferentes optimizadores.

El tiempo de ejecución del algoritmo propuesto (Tabla 4) se registra en minutos. Se evalúan las variaciones del mismo con respecto al uso de diferentes optimizadores y con respecto al incremento del número de iteraciones en los algoritmos metaheurísticos.

4.2.2. Conjuntos de datos

El procedimiento de optimización es aplicado en diversos conjuntos de datos a fin de tener la mayor cantidad posible de resultados experimentales, pero además permite verificar que el algoritmo propuesto sea replicable y escalable. Los conjuntos de datos son extraídos de fuentes de datos abiertos en internet. La principal fuente a la que se ha recurrido es el repositorio de aprendizaje automático de la Universidad de Carolina de Irvine (Bache & Lichman, 2013) que contiene una colección de bases de datos utilizadas para el análisis empírico de algoritmos de aprendizaje automático. Asimismo, se utilizan conjuntos de datos divulgados en publicaciones académicas.

A. Datos sobre rendimiento académico

El conjunto contiene datos divulgados por Cortez y Silva (2008) sobre trescientos noventa y cinco (395) estudiantes de dos escuelas secundarias portuguesas, en Matemáticas y Portugués (Bache & Lichman, 2013). Mide cuarenta y seis (46) variables: calificaciones en tres periodos, G1, G2 y G3, características demográficas, sociales y otras relacionadas con la escuela, todas recopiladas mediante informes y cuestionarios escolares. Cortez y Silva (2008) modelaron los datos en tareas de clasificación y regresión ordinal univariada, tomando exclusivamente la última calificación G3 como variable de respuesta, y se contrastaron los resultados, de cuando se incluyen como variables predictoras G2 y G1, y cuando son excluidas. Se ha observado que también deben explorarse los métodos de selección de características, ya que solo una pequeña parte de las variables de entrada consideradas parecen ser pertinentes.

B. Datos sobre enfermedad de Parkinson

El conjunto de datos divulgado por Tsanas et al. (2009) se compone de mediciones biomédicas de la voz de 42 personas con enfermedad de Parkinson en etapa inicial reclutados para el monitoreo remoto de la progresión de los síntomas (Bache & Lichman, 2013). Las variables predictoras corresponden al número de sujeto, edad del sujeto, género del sujeto, intervalo de tiempo desde la fecha de reclutamiento inicial y 16 medidas biomédicas de voz. Los datos son utilizados en tareas de predicción de los puntajes en la Escala Unificada de Calificación de la Enfermedad de Parkinson UPDRS, motores y totales (Little et al., 2008;

Tsanas et al., 2009). Esta escala se utiliza para hacer un seguimiento de la enfermedad de Parkinson, pero es costosa y logísticamente inconveniente (Benmalek et al., 2015).

Varios estudios de investigación han utilizado diferentes técnicas para intentar predecir la gravedad de los síntomas de la enfermedad de Parkinson a partir de las pistas del habla (Sakar et al., 2013). En particular, en el mismo conjunto de datos, se aplicaron métodos de selección de características (Tsanas et al., 2009), técnicas de regresión por mínimos cuadrados, clasificación no paramétrica y árboles de regresión. Benmalek et al., (2015) probaron redes neuronales artificiales. En ambos trabajos, a pesar de los buenos resultados, se destacó la alta correlación entre algunas medidas de disfonía, y se sugirió explorar más a fondo los métodos no lineales

C. Otros conjuntos de datos utilizados

Son varios los conjuntos de datos utilizados en la etapa de diseño del algoritmo. No se realizan extensas evaluaciones ni comparaciones estadísticas sobre los mismos y su uso contribuye a diagnosticar la codificación informática y realizar ajustes.

- Conjunto de datos divulgado por Yeh (2007) constituido por 103 (ciento tres) registros de pruebas sobre concreto de alto rendimiento (HPC) en 7 (siete) variables independientes: cemento, escoria, ceniza volante, agua, súper plastificante, agregado grueso y agregado fino y 3 (tres) variables de respuesta: asentamiento, flujo y compresión a los 28 días (Bache & Lichman, 2013). Los datos son utilizados fundamentalmente en la construcción de modelos predictivos univariados, como redes neuronales y modelos lineales (Yeh, 2007; Agrawal & Sharma, 2010; Chandwani et al., 2014)
- Conjunto de datos divulgado por Harshman et al. (1977) consistente en 13 medidas pseudo sagitales obtenidas de cinco oradores en la pronunciación de diez sonidos vocales en inglés. Los datos se obtienen a partir de grabaciones de audio de alta calidad y cine-fluorogramas. Las formas de la lengua se caracterizan midiendo la distancia a lo largo de líneas de referencia entre la superficie de la lengua y 17 puntos predeterminados. Harshman et al. (1977) realizan un análisis factorial de los datos.
- Conjuntos de datos “cornell” y “carscomplete” incluidos en el paquete *plsdepot* (Sánchez, 2012). El conjunto de datos “cornell” contiene observaciones sobre ocho

variables que describen la composición de doce gasolinas con diferente octanaje, mientras que "carscomplete" contiene observaciones sobre seis variables que describen las características de veinticuatro vehículos.

Los ensayos realizados sobre el conjunto de datos "cornell" (Sánchez, 2012) y el conjunto de datos sobre concreto de alto rendimiento (Yeh, 2007) ponen a prueba el método de optimización con varias funciones kernel. Los resultados se presentan en la sección de Anexos A.3.

4.3. Herramientas computacionales

Uno de los objetivos de la tesis doctoral es la generación de herramientas computacionales para calibrar modelos de regresión KPLS con fines predictivos. En esta sección se presenta el soporte del software R, los paquetes y funciones utilizados para el desarrollo de las evaluaciones experimentales.

Como resultado de las evaluaciones experimentales y la mejora continua de la codificación, se crea un paquete de implementación en el software R publicado en el repositorio GitHub denominado *optimKPLS* que integra una función para la selección de valores aproximados del parámetro de la función kernel y el número de componentes.

4.3.1. Soporte del software R

El algoritmo de selección de parámetros de la regresión KPLS (Tabla 4) está programado en el software R y para su ejecución es necesaria la instalación previa de una combinación de paquetes:

a) *plsdepot* de Sánchez (2012) y *kernlab* de Zeileis et al. (2004) para la ejecución de la regresión KPLS
b) *metaheuristicOpt* de Riza y Nugroho (2018), *dfoptim* de Varadhan y Borchers (2016) o *Rmalschains* de Bergmeir et al. (2012) para la elección de optimizadores.

Con este diseño es posible realizar varias configuraciones del algoritmo para su experimentación, seleccionar funciones kernel y sus parámetros, número máximo de

componentes y especificaciones de los optimizadores. Los códigos (scripts) se hallan disponibles en la sección de Anexos A.1.

A. *plsdepot.* Partial Least Squares (PLS) Data Analysis Methods

Este paquete contiene diferentes métodos para el análisis PLS de una o dos tablas de datos, como NIPALS, SIMPLS, Regresión PLS, Análisis canónico PLS, entre otros (Sánchez, 2012). La referencia principal para este software es el libro "La Regression PLS: Theorie et Pratique" de Michel Tenenhaus (1998). De este paquete se utiliza principalmente la función `plsreg2`. Esta función realiza la regresión de mínimos cuadrados parciales para el caso multivariado. Su uso y argumentos se presentan a continuación:

```
plsreg2(predictors, responses, comps, crosval)
```

El argumento `predictors` contiene la matriz numérica o marco de datos de las variables de predicción, `responses` contiene la matriz numérica o marco de datos correspondientes a las variables de respuesta, `comps` representa el número de componentes extraídos (por defecto su valor es 2) y el argumento `crosval` es un valor lógico que indica si se realiza el procedimiento de validación cruzada (`TRUE` por defecto).

B. *kernlab.* Kernel-Based Machine Learning Lab

Este paquete contiene métodos de aprendizaje automático basados en kernel para clasificación, regresión, agrupación, reducción de la dimensionalidad, entre otros. Incluye máquinas de soporte vectorial (SVM) y Kernel PCA (Zeileis et al., 2004). De este paquete se utiliza la función `kernelMatrix` para el cálculo de la matriz Kernel. Su uso y argumentos son de la siguiente manera:

```
kernelMatrix(kernel, x)
```

Donde el argumento `kernel` representa la función kernel que se utilizará para calcular la matriz kernel. El paquete dispone las funciones kernel más populares como `rbfdot` para la función kernel gaussiana, `polydot` para la función kernel polinomial, `laplacedot` para la función kernel laplaciana, entre varias otras. El argumento `x` contiene la matriz de datos que se utilizará para calcular la matriz kernel.

C. *metaheuristicOpt.* Metaheuristic for Optimization

Este paquete contiene implementaciones de 11 algoritmos metaheurísticos para optimización continua. Entre ellos GA, PSO, FFA y GWO (Riza & Nugroho, 2018). La función principal para ejecutar los algoritmos es `metaOpt`. Su uso y argumentos se presentan a continuación:

```
metaOpt(FUN, optimType, algorithm, numVar, rangeVar,
      control = list())
```

Donde el argumento `FUN` constituye la función objetivo, `optimType` representa el tipo de optimización, con las opciones `"MIN"` y `"MAX"` para problemas de minimización y maximización, respectivamente. El argumento `algorithm` representa el algoritmo utilizado para hacer la optimización. Actualmente hay once algoritmos implementados, entre ellos: `"GA"` Algoritmos genéticos, `"FFA"` Algoritmo de luciérnagas, `"PSO"` Optimización de enjambre de partículas `y "GWO"`: Optimización basado en el comportamiento de lobos grises.

El número y rango de las variables se determina por los argumentos `numVar` y `rangeVar`. El argumento `control` es una lista que contiene los parámetros necesarios para cada algoritmo:

- GA: `list(numPopulation, maxIter, Pm, Pc)`
- PSO: `list(numPopulation, maxIter, Vmax, ci, cg, w)`
- FFA: `list(numPopulation, maxIter, B0, gamma, alpha)`
- GWO: `list(numPopulation, maxIter)`

D. *Rmalschain:* Continuous Optimization using Memetic Algorithms with Local Search Chains.

Este paquete implementa una familia de algoritmos meméticos, hibridaciones de algoritmos genéticos con métodos de búsqueda local. Son especialmente adecuados para la optimización continua. La función utilizada es `malschains`. Su uso y argumentos fundamentales se presentan a continuación:

```
malschains(fn, lower, upper, dim, maxEvals, verbosity, initialpop,
    control = malschains.control(),seed, env)
```

Donde el argumento `fn` constituye la función objetivo y `maxEvals` representa el número de evaluaciones de la función objetivo, que en esta investigación se corresponde al número de iteraciones. Con la función `malschains.control()` es posible definir el tamaño de población del algoritmo genético `popsize` así como otros parámetros.

E. dfoptim. Derivative-Free Optimization

Este paquete implementa algoritmos de optimización que no requieren información de gradientes. Actualmente están implementados los algoritmos Nelder-Mead y Hooke-Jeeves. (Varadhan & Borchers, 2016). Los usos y argumentos de estos algoritmos son:

- `nmk(par, fn, control = list(), ...)` para el algoritmo Nelder-Mead
- `hjk(par, fn, control = list(), ...)` para el algoritmo Hooke-Jeeves

Donde `par` es el vector inicial de valores de los parámetros y `fn` es la función objetivo no lineal a ser optimizada. El argumento `control` es una lista que especifica cambios en parámetros predeterminados de los algoritmos. Uno de los elementos de la lista es `tol`, que indica el criterio de convergencia, es decir, la iteración concluye cuando la longitud de paso del bucle principal es menor que `tol`. Su valor por defecto es `1.e-06`.

4.3.2. Codificación y documentación

Se crea el paquete *optimKPLS* con una función de calibración de modelos de regresión KPLS, para la selección de parámetros de la función kernel y el número de componentes de la regresión. Toma algoritmos meméticos MMA como optimizador iterativo de la función.

El paquete *optimKPLS* se comparte en el repositorio digital GitHub en el siguiente enlace https://github.com/jorgemellopy/optimKPLS. La documentación se describe a continuación:

```
Package           : optimKPLS
Type              : Package
Title             : Optimización de KPLS mediante un algoritmo memético
Version           : 0.0.1.0
```

```
Authors@R         : c(person("Jorge Daniel Mello Roman", "Developer",
                  role = c("aut", "cre"), email = "jmello@ucm.es"))
Maintainer        : Jorge Daniel Mello Roman <jmello@ucm.es>
Description       : Ajuste metaheurístico de parámetros de la regresión
                  KPLS.
Depends           : R (>= 3.5.0)
Imports           : kernlab,plsdepot,Rmalschains,readxl
License           : GPL-3
Encoding          : UTF-8
LazyData          : true
VignetteBuilder   : knitr
RoxygenNote       : 7.1.1
```

La única función del paquete es `optim_kpls` y retorna la estimación puntual del parámetro σ de la función kernel gaussiana, el número de componentes *h* y el valor de la función objetivo Q^2_{cum}.

```
<- optim_kpls (data1,data2,numCompoments,lo,up,po,mE)
```

Donde `data1` es el conjunto de datos de variables independientes, `data2` es conjunto de datos de variables dependientes, `numCompoments` el número máximo de componentes, `lo,up` son los límites inferior y superior del parámetro `po`, es el tamaño de la población y `mE` el número máximo de evaluaciones de la función objetivo.

El paquete *optimKPLS* se encuentra en una etapa inicial de desarrollo, no participa en las evaluaciones experimentales que se presentan en este documento y se obtiene como corolario de las conclusiones de la investigación doctoral y la mejora continua en la documentación de los códigos. La codificación del paquete se detalla en el Anexo A.2.

Capítulo 5

Evaluaciones experimentales

Las evaluaciones y comparaciones experimentales se presentan en este capítulo. Se organiza en tres secciones que describen procesos independientes pero orientados a la solución del mismo problema de optimización. La secuencialidad observada refleja el proceso de aprendizaje de la problemática y el desarrollo de capacidades para su abordaje.

5.1. Optimización con algoritmos genéticos

En esta sección se presenta un resumen de la metodología implementada y los resultados de la optimización de KPLS mediante GA.

5.1.1. Metodología

La programación en el software R requirió la instalación de los paquetes: *plsdepot*, *kernlab* y *dfotimp* descritos en la Sección 4.3.1. Se utilizó el conjunto de datos sobre rendimiento académico presentado en la Sección 4.2.2. Fueron definidas como variables de respuesta **Y** las tres (3) puntuaciones G1, G2 y G3 obtenidas por los estudiantes en la asignatura "Matemáticas". Las variables predictoras X constituyen las cuarenta y tres (43) variables restantes del conjunto de datos.

Tomando como referencia otras investigaciones, se seleccionó la función kernel gaussiana, que en el paquete *kernlab* se define:

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\sigma\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \quad (23)$$

Siendo el parámetro de la función kernel $\theta = \sigma > 0$ configurado como individuo del optimizador GA. El procedimiento de optimización ejecuta los pasos descritos en la Tabla 4. Se realizaron treinta $k = 30$ ejecuciones extrayendo para cada una de ellas:

- el valor máximo Q^2_{cum}
- los valores de σ y h asociados al valor máximo de Q^2_{cum}
- la dispersión en las estimaciones de σ y h determinada por la Ecuación 22
- el tiempo de ejecución en minutos

Las especificaciones (hiperparámetros) de GA fueron establecidas con referencia a otras investigaciones (Noorizadeh et al., 2011; Brownlee, 2011): selección por ruleta, punto de cruce a partir de la quinta posición, probabilidad de mutación > 10% y reemplazo aleatorio.

Se realizaron pruebas preliminares para determinar el tamaño de la población y el número máximo de generaciones. Las pruebas se llevaron a cabo en paralelo de la siguiente manera:

- Incremento gradual del número de generaciones $g = \{10,20,30,40,50,100,200\}$ manteniendo constante el tamaño de la población $p = 20$.
- Incremento gradual del tamaño de la población $p = \{20,40,60,80,100,200,400\}$ con un número de generaciones $g = 10$ constante.

El criterio de convergencia establecido fue seleccionar los valores del número de generaciones g y del tamaño de la población p para los cuales la diferencia en la dispersión relativa con respecto a la prueba anterior, tanto para D_σ como D_h sean inferiores a un umbral fijado en 0,05 = 5%.

Para verificar los resultados también se evaluó el rendimiento del algoritmo Hooke-Jeeves (HJ) como optimizador. Las ejecuciones se realizaron estableciendo aleatoriamente 30 diferentes puntos de partida $\sigma_0 > 0$. En cada ejecución las iteraciones de HJ concluyeron cuando la longitud de paso del bucle principal se hizo más pequeña que $1e^{-0}$.

Finalmente, se aplicó el PLS simple al mismo conjunto de datos a fin de comparar los resultados finales obtenidos por la regresión KPLS optimizada con GA y HJ.

5.1.2. Resultados

Los resultados de las pruebas preliminares para determinar el tamaño de la población p y el número máximo de generaciones g se presentan en la Tabla 5. Las características del ordenador donde se realizaron las pruebas son: Intel ® Celeron ® CPUG1610 @ 2.600GHz, 4.00 GB RAM.

Tabla 5. Pruebas preliminares para determinar p y g en algoritmos genéticos (GA)

	Indicadores	Número de generaciones						
		$g = 10$	$g = 20$	$g = 30$	$g = 40$	$g = 50$	$g = 100$	$g = 200$
Tamaño de la población $p = 20$	Max Q^2_{cum}	0,9997	0,9998	0,9998	0,9998	0,9998	0,9998	0,9998
	σ	4,276	4,124	4,074	4,024	4,179	4,224	4,269
	h	7	7	7	7	7	7	7
	D_σ	27,18%	21,24%	17,67%	14,10%	13,38%	12%	10,41%
	D_h	16,08%	13,80%	12,59%	11,37%	9,40%	7,90%	6,39%
	Tiempo medio (min.)	21,03	41,81	62,48	81,48	110,40	210,42	420,85
	Indicadores	Tamaño de la población						
Número de generaciones $g = 10$		$p = 20$	$p = 40$	$p = 60$	$p = 80$	$p = 100$	$p = 200$	$p = 400$
	Max Q^2_{cum}	0,9997	0,9997	0,9998	0,9998	0,9998	0,9998	0,9998
	σ	4,276	4,116	4,096	4,097	4,099	4,120	4,142
	h	7	7	7	7	7	7	7
	D_σ	27,18%	7,91%	4,44%	4,26%	4,07%	3,92%	3,76%
	D_h	16,08%	7,49%	6,39%	5,46%	4,52%	4,11%	3,69%
	Tiempo medio (min.)	21,03	45,09	67,34	88,03	114,47	221,86	443,69

La Tabla 5 muestra que el valor máximo de Q^2_{cum} se alcanza con la primera evaluación del procedimiento y permanece constante con el aumento tanto del número de generaciones como del tamaño de la población.

La dispersión relativa, tanto en las estimaciones del parámetro de la función kernel D_σ como en las estimaciones del número de componentes D_σ, se redujo gradualmente con el aumento del número de generaciones.

Se observó una convergencia inmediata en las estimaciones con el aumento del tamaño de la población. El tiempo de ejecución medio de cada procedimiento tiene una relación lineal directa con el número de generaciones y el tamaño de la población de GA.

De acuerdo con el criterio de convergencia establecido en términos de reducción de la dispersión relativa, para este conjunto de datos y la función kernel gaussiana, el tamaño de la población se determinó en $p = 60$ y el número de generaciones $g = 30$.

Con estos valores de p y g, se realizaron otras treinta (30) ejecuciones del procedimiento de optimización. El valor máximo alcanzado por el coeficiente Q^2_{cum} fue de 0,9998. El parámetro de la función kernel gaussiana σ = 4,1710 con$D_\sigma = 3.78\%$ y el número de componentes h = 7 con $D_h = 3.61\%$ fueron los valores asociados al óptimo Q^2_{cum}.

En cuanto al conjunto de datos sobre rendimiento académico en Matemáticas, los resultados indican que (7) siete componentes extraídos de (43) variables predictoras tienen la capacidad de predecir en 0,9998 = 99,98% los valores de (3) tres variables de respuesta G1, G2 y G3, cuando el parámetro de la función kernel σ = 4,1710.

Para comprobar los valores obtenidos por GA, se evaluó el rendimiento del algoritmo Hooke-Jeeves. En la Tabla 6 se muestran los resultados obtenidos por la regresión KPLS optimizada por ambos algoritmos. También compara los resultados obtenidos por la regresión PLS simple, en el mismo conjunto de datos.

Tabla 6. Resultados de la regresión KPLS optimizada con GA y HJ y la regresión PLS.

	Max Q^2_{cum}	σ	h	D_σ	D_h
GA - KPLS	0,9998	4,1710	7	3,78%	3,61%
HJ - KPLS	0,9996	4,0967	7	38,43%	45,08%
PLS	0,1466	-	1	-	-

El algoritmo HJ obtuvo valores similares a GA para el valor máximo de Q^2_{cum}, σ y h. Sin embargo, las variaciones en las estimaciones D_σ y D_h fueron superiores a las arrojadas por GA. Se observó además que las soluciones de HJ fueron atrapadas con frecuencia en aparentes óptimos locales.

La regresión PLS también se aplicó al mismo conjunto de datos. El valor más alto alcanzado por PLS fue $Q^2_{cum} = 0{,}1466 = 14{,}66\%$ extrayendo un h = 1 componente. En términos de la capacidad predictiva de la regresión, este resultado es muy inferior al obtenido por KPLS con la función kernel gaussiana y los valores de los parámetros: σ = 4,1710 y h = 7.

5.2. Optimización con algoritmos inspirados en la naturaleza

En esta sección se presenta un resumen de la metodología implementada y los resultados de la optimización de KPLS mediante GA, PSO, GWO y FFA.

5.2.1. Metodología

La programación en el software R requirió la instalación de los paquetes: *plsdepot*, *kernlab*, *metaheuristicOpt*, *dfoptim* presentados en la Sección 4.3.1. Fue seleccionada la función kernel gaussiana (Ecuación 23).

Se realizaron pruebas preliminares para seleccionar los hiperparámetros de los algoritmos metaheurísticos inspirados en la naturaleza. Se evaluaron varios conjuntos de hiperparámetros manteniendo constantes tanto el tamaño de la población como el número de iteraciones. Fueron seleccionados los conjuntos de hiperparámetros para los cuales se obtuvieron valores medios más altos de Q^2_{cum} y con menor dispersión.

Definidos los hiperparámetros de los algoritmos metaheurísticos, se realizaron k = 30 ejecuciones el algoritmo de selección de parámetros de la regresión KPLS (Tabla 4). Se extrajo por cada ejecución: el valor máximo de Q^2_{cum} y sus valores asociados tanto del número de componentes h como del parámetro de la función kernel gaussiana σ para cada ejecución. Además, el tiempo de ejecución se calculó en minutos.

A fin de evaluar la convergencia de las estimaciones se repitió el proceso para un conjunto de iteraciones $g = \{10,30,50,100,150\}$ manteniendo el constante del tamaño de la población en el valor $p = 40$.

Para validar los resultados y tener un punto de referencia de la eficiencia de los optimizadores, se implementaron los algoritmos HJ y NM. Por cada algoritmo se realizaron $k = 30$ ejecuciones con puntos de partida inicial aleatorios. Los hiperparámetros de HJ y NM se establecieron con referencia a otras investigaciones.

Se realizó un análisis estadístico para comprobar la hipótesis de que no hay diferencias estadísticamente significativas entre las estimaciones de los distintos algoritmos metaheurísticos. El ANOVA se ejecutó tomando como hipótesis nula que la media de las estimaciones Q^2_{cum},σ y h eran similares para los diferentes algoritmos evaluados. Se complementó con pruebas de homogeneidad de la varianza y comparaciones múltiples post hoc para identificar los algoritmos de mejor rendimiento. La prueba T2 Tamhane se usó para comparaciones post hoc. Además, se aplicó la prueba de normalidad de Kolmogorov-Smirnov sobre la distribución de las estimaciones de σ.

Se utilizaron dos conjuntos de datos presentados en la Sección 4.2.2, el primero contiene mediciones biomédicas de la voz en personas con la enfermedad de Parkinson, y el segundo, datos sobre rendimiento académico en Matemáticas.

5.2.2. Resultados

Se realizaron pruebas preliminares para seleccionar los hiperparámetros de los algoritmos metaheurísticos para los dos conjuntos de datos utilizados. Los resultados se presentan en la Tabla 7.

Se realizaron $k = 30$ ejecuciones del algoritmo de selección de parámetros (Tabla 4) utilizando como optimizadores las metaheurísticas GA, FFA, PSO y GWO y los algoritmos de búsqueda directa NM y HJ. Para generar la matriz kernel **K**, se utilizó la función kernel gaussiana con parámetro $\sigma > 0$.

El tamaño de la población de los algoritmos metaheurísticos se fijó en $p = 40$, y las iteraciones se aumentaron gradualmente por $g = \{10,30,50,100,150\}$. Los algoritmos NM y HJ

tomaron 30 puntos de partida aleatorios $\sigma_0 > 0$ para ambos conjuntos de datos. El criterio de convergencia fue el valor $1e^{-0}$.

Tabla 7. Hiperparámetros de algoritmos metaheurísticos inspirados por la naturaleza por conjunto de datos.

Algoritmos	Hiperparámetros	Conjunto de datos	
		Enfermedad de Parkinson	Rendimiento académico
PSO	V_{max} velocidad máxima	2	4
	C_1 aprendizaje individual	2	2
	C_2 aprendizaje colectivo	2	2
	ω factor de inercia	0,729	0,729
FFA	β_0 coeficiente de atracción	0.6	1
	γ coeficiente de absorción	1	1
	α parámetro de aleatorización	0,5	0,5
GA	P_m probabilidad de mutación	0,5	0,5
	P_c probabilidad de cruce	0,8	0,5
GWO	lb límite inferior	0	0
	ub límite superior	100	10

En las Figuras 2 y 3 se presenta el comportamiento de las estimaciones de Q^2_{cum} por cada conjunto de datos. Los algoritmos metaheurísticos estimaron valores promedio más altos de Q^2_{cum} y con una menor dispersión que los algoritmos determinísticos. Las estimaciones de Q^2_{cum} fueron muy similares para los cuatro algoritmos metaheurísticos, con variaciones mínimas observadas con el creciente número de iteraciones.

A pesar de las diferencias mencionadas, las estimaciones de los algoritmos determinísticos no difieren considerablemente de los valores obtenidos por los algoritmos metaheurísticos. Los detalles pueden verse en las Tablas 8 y 9.

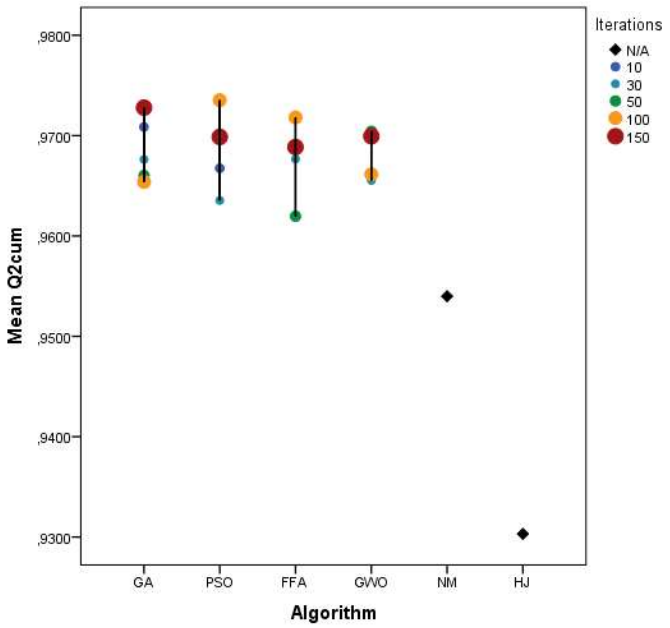
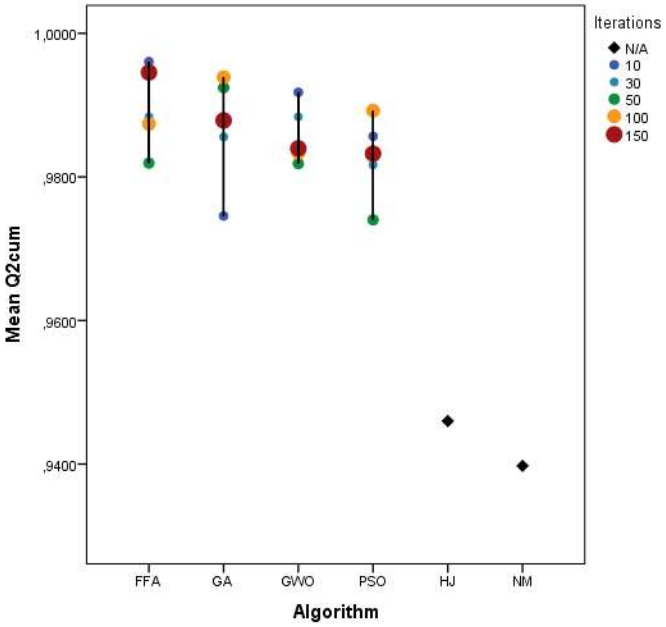


Figura 2. Media de estimaciones de Q^2_{cum} por optimizador. Datos sobre enfermedad de Parkinson



.

Figura 3. Media de estimaciones de Q^2_{cum} por optimizador. Datos sobre rendimiento académico

Tabla 8. Media y desviación estándar de las estimaciones de Q^2_{cum}, σ, h y tiempo de ejecución. Datos sobre enfermedad de Parkinson.

		Algoritmos											
		GA		PSO		FFA		GWO		NM		HJ	
	g	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.
Q^2_{cum}	10	0,971	0,016	0,967	0,020	0,969	0,016	0,970	0,015	.	.	.	.
	30	0,968	0,017	0,964	0,019	0,968	0,017	0,966	0,019	.	.	.	.
	50	0,966	0,017	0,970	0,015	0,962	0,020	0,970	0,013	.	.	.	.
	100	0,965	0,018	0,974	0,010	0,972	0,013	0,966	0,018	.	.	.	.
	150	0,973	0,012	0,970	0,017	0,969	0,015	0,970	0,013	.	.	.	.
	N/A	.	.	.	.	.	.	.	.	0,954	0,045	0,930	0,049
σ	10	49,17	3,32	47,43	3,49	48,67	2,80	48,13	3,33	.	.	.	.
	30	49,00	3,31	47,35	2,99	47,42	3,30	47,04	3,15	.	.	.	.
	50	48,83	3,27	47,06	2,30	47,10	3,23	47,20	2,73	.	.	.	.
	100	48,58	2,74	47,51	2,95	48,14	2,68	47,18	3,12	.	.	.	.
	150	48,44	2,57	47,09	2,49	48,68	3,12	45,27	3,01	.	.	.	.
	N/A	.	.	.	.	.	.	.	.	40,86	14,20	51,53	27,17
h	10	6	0	6	0	6	0	6	0	.	.	.	.
	30	6	0	6	0	6	0	6	0	.	.	.	.
	50	6	0	6	0	6	0	6	0	.	.	.	.
	100	6	0	6	0	6	0	6	0	.	.	.	.
	150	6	0	6	0	6	0	6	0	.	.	.	.
	N/A	.	.	.	.	.	.	.	.	6	0	6	0
Tiempo (min.)	10	124,3	4,6	119,7	3,1	70,6	2,9	71,1	3,8	.	.	.	.
	30	352,9	22,5	513,6	92,8	201,5	14,0	208,1	4,0	.	.	.	.
	50	571,2	55,8	800,6	117,2	332,9	27,7	334,8	19,6	.	.	.	.
	100	1135,4	80,7	1141,8	474,4	616,6	59,5	664,9	32,5	.	.	.	.
	150	1509,7	347,1	1734,7	727,1	940,4	83,7	852,0	156,4	.	.	.	.
	N/A	.	.	.	.	.	.	.	.	5,1	0,9	18,0	2,6

Tabla 9. Media y desviación estándar de las estimaciones de Q^2_{cum}, σ, h y tiempo de ejecución. Datos sobre rendimiento académico.

		Algoritmos											
		GA		PSO		FFA		GWO		NM		HJ	
	g	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.	Media	Desv. Est.
Q^2_{cum}	10	0,975	0,056	0,986	0,045	0,996	0,003	0,992	0,032	.	.	.	.
	30	0,986	0,041	0,982	0,047	0,988	0,035	0,988	0,036	.	.	.	.
	50	0,992	0,026	0,974	0,061	0,982	0,047	0,982	0,049	.	.	.	.
	100	0,994	0,019	0,989	0,031	0,987	0,035	0,983	0,050	.	.	.	.
	150	0,988	0,034	0,983	0,045	0,995	0,003	0,984	0,041	.	.	.	.
	N/A	.	.	.	.	.	.	.	.	0,940	0,220	0,946	0,210
σ	10	3,90	0,65	4,07	0,21	3,86	0,71	4,11	0,13	.	.	.	.
	30	4,08	0,37	4,16	0,13	4,05	0,55	4,08	0,16	.	.	.	.
	50	4,04	0,39	4,07	0,16	3,98	0,75	4,11	0,12	.	.	.	.
	100	3,87	0,66	4,06	0,14	3,73	0,82	4,08	0,17	.	.	.	.
	150	3,70	0,82	4,11	0,13	4,02	0,58	4,09	0,13	.	.	.	.
	N/A	.	.	.	.	.	.	.	.	7,11	4,96	6,01	2,89
h	10	7,4	0,7	7,1	0,4	7,2	0,8	7,1	0,3	.	.	.	.
	30	7,2	0,6	7,0	0,3	7,1	0,8	7,1	0,4	.	.	.	.
	50	7,2	0,5	7,2	0,4	7,1	1,1	7,0	0,2	.	.	.	.
	100	7,4	0,7	7,1	0,4	7,4	1,0	7,1	0,4	.	.	.	.
	150	7,5	0,9	7,1	0,3	7,1	0,8	7,1	0,3	.	.	.	.
	N/A	.	.	.	.	.	.	.	.	6,5	1,4	6,4	1,3
Tiempo (min)	10	56,9	4,1	109,5	6,1	32,0	1,8	38,3	2,7	.	.	.	.
	30	155,8	8,5	274,2	5,5	90,9	3,5	99,5	8,1	.	.	.	.
	50	261,2	12,6	483,0	21,3	145,2	8,3	146,3	21,3	.	.	.	.
	100	467,2	60,7	822,8	106,7	274,6	20,7	236,1	26,6	.	.	.	.
	150	576,1	123,3	978,7	395,7	368,1	79,5	304,6	93,1	.	.	.	.
	N/A	.	.	.	.	.	.	.	.	1,8	0,4	7,4	0,9

Se realizó el análisis estadístico de las estimaciones de Q^2_{cum}. Se ejecutó el ANOVA tomando como hipótesis nula la igualdad de medias de Q^2_{cum} para los diferentes optimizadores. El análisis se complementó con una prueba de homogeneidad de la varianza. Los resultados presentados en la Tabla 10 indican que existe evidencia estadística para rechazar la hipótesis nula. El resultado arrojado por la prueba de Levene indica la no homogeneidad de las varianzas.

Las estimaciones del parámetro de la función kernel σ fueron en promedio similares para los algoritmos evaluados en ambos conjuntos de datos. Los detalles pueden verse en las Tablas 8 y 9. NM y HJ mostraron valores medios de σ cercanos a los obtenidos por los algoritmos metaheurísticos, pero con una desviación estándar mucho mayor.

En la Tabla 10 se muestran los resultados del ANOVA realizado para contrastar la hipótesis nula de igualdad de medias de σ para los diferentes optimizadores. El test ANOVA indica que existe evidencia estadística para rechazar la hipótesis nula un nivel de significación α=0.05. De la misma manera, la prueba de Levene indica la no homogeneidad de las varianzas.

Tabla 10. ANOVA y prueba de homogeneidad de varianzas de Q^2_{cum} y σ por algoritmo.

	Conjunto de datos	ANOVA		Homogeneidad de varianzas	
		F	Sig.	Estad. de Levene	Sig.
Q^2_{cum}	Enfermedad de Parkinson	25,453	0,000	25,788	0,000
	Rendimiento académico	3,247	0,007	14,079	0,000
σ	Enfermedad de Parkinson	6,329	0,000	169,87	0,000
	Rendimiento académico	2,793	0,017	21,905	0,000

Para las comparaciones múltiples post hoc, se utilizó la prueba T2 Tamhane. En el conjunto de datos sobre la enfermedad de Parkinson los resultados indican una diferencia significativa entre las estimaciones Q^2_{cum} del algoritmo HJ con respecto a los demás algoritmos. En el conjunto de datos sobre el rendimiento académico, las comparaciones por pares no fueron significativas, mientras que el efecto global fue débilmente significativo (Ver Tabla 10). Este resultado podría justificarse tanto por el débil efecto global como por las características conservadoras de la prueba T2 Tamhane. Los resultados se presentan en la Tabla 11.

Tabla 11. Prueba T2 Tamhane de comparaciones múltiples – Q^2_{cum} por algoritmo

Algoritmos		Datos sobre enfermedad de Parkinson			Datos sobre rendimiento académico		
		Diferencia de medias*	Error Est.	Sig.	Diferencia de medias*	Error Est.	Sig.
GA	PSO	0,0000	0,0020	1,0000	0,0042	0,0050	0,9990
GA	FFA	0,0007	0,0019	1,0000	0,0026	0,0041	1,0000
GA	GWO	0,0001	0,0019	1,0000	0,0010	0,0046	1,0000
GA	NM	0,0145	0,0095	0,8940	0,0472	0,0551	1,0000
GA	HJ	0,0382	0,0077	0,0000	0,0410	0,0484	1,0000
PSO	FFA	0,0007	0,0020	1,0000	0,0069	0,0047	0,9010
PSO	GWO	0,0001	0,0019	1,0000	0,0032	0,0052	1,0000
PSO	NM	0,0145	0,0095	0,8940	0,0430	0,0551	1,0000
PSO	HJ	0,0382	0,0077	0,0000	0,0367	0,0484	1,0000
FFA	GWO	0,0006	0,0019	1,0000	0,0036	0,0043	1,0000
FFA	NM	0,0138	0,0095	0,9250	0,0498	0,0550	0,9990
FFA	HJ	0,0375	0,0077	0,0000	0,0436	0,0483	0,9990
GWO	NM	0,0144	0,0095	0,8980	0,0462	0,0551	1,0000
GWO	HJ	0,0381	0,0077	0,0000	0,0399	0,0484	1,0000
NM	HJ	0,0237	0,0121	0,574	0,0062	0,0732	1,0000

*Diferencia de medias en valor absoluto

El número de componentes *h* estimado fue igual a 6 para todos los algoritmos evaluados en el conjunto de datos sobre la enfermedad de Parkinson. En el caso del conjunto de datos sobre el rendimiento académico, las estimaciones fueron variables; sin embargo, la estimación más frecuente para todos los algoritmos fue *h* = 7. Los resultados se presentan en la Tabla 12.

Tabla 12. Estimaciones de h para el conjunto de datos sobre rendimiento académico por algoritmo.

h	Algoritmo					
	FFA	GA	GWO	PSO	HJ	NM
5	0,7%	0,0%	0,0%	0,0%	42,1%	37,5%
6	18,1%	2,1%	0,7%	0,7%	0,0%	0,0%
7	54,3%	73,4%	89,7%	86,8%	47,4%	50,0%
8	16,7%	14,0%	9,6%	12,5%	0,0%	0,0%
9	10,1%	10,5%	0,0%	0,0%	10,5%	12,5%

Un resultado favorable a la estimación en intervalo del parámetro σ es la distribución aproximadamente normal de las estimaciones de ciertos algoritmos. Tomando $h = 6$ para el conjunto de datos sobre la enfermedad de Parkinson y $h = 7$ para el conjunto de datos de rendimiento académico, se aplicó la prueba de normalidad Kolmogorov – Smirnov, bajo la hipótesis nula de la distribución normal de las estimaciones de σ.

A un nivel de significación $\alpha = 0,05$, no se rechazó la hipótesis para los algoritmos PSO y GWO en ambos conjuntos de datos. Los detalles se muestran en la Tabla 13. En las Figuras 4 y 5 se visualizan las distribuciones de las estimaciones de σ por algoritmo metaheurístico, y para cada conjunto de datos.

Tabla 13. Prueba de normalidad Kolmogorov-Smirnov para la distribución de σ

Datos de la enfermedad de Parkinson $h = 6$			Datos de rendimiento académico $h = 7$		
Algoritmo	Estadístico	Sig.	Algoritmo	Estadístico	Sig.
FFA	0,074	0,048	FFA	0,130	0,003
GA	0,044	0,200	GA	0,100	0,011
GWO	0,063	0,200	GWO	0,050	0,200
PSO	0,037	0,200	PSO	0,053	0,200
HJ	0,092	0,200	HJ	0,275	0,048
NM	0,274	0,000	NM	0,369	0,002

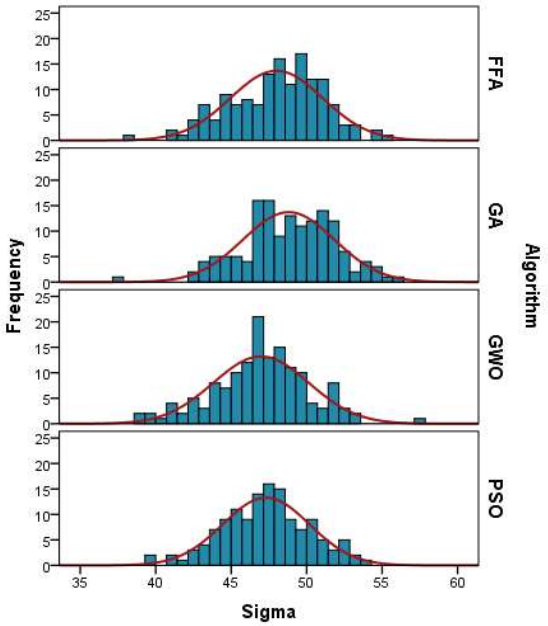


Figura 4. Distribución de estimaciones σ por algoritmo metaheurístico. Datos de la enfermedad de Parkinson

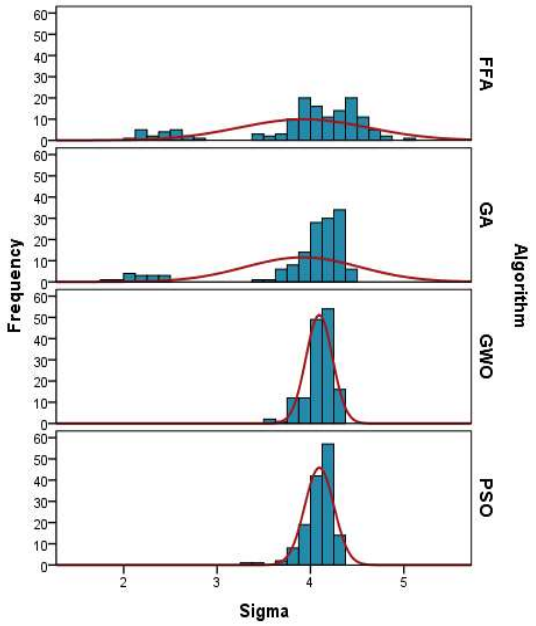


Figura 5. Distribución de estimaciones σ por algoritmo metaheurístico. Datos de rendimiento académico

Se consideran determinados los valores óptimos de Q^2_{cum}, h y σ. Según García-Ródenas et al. (2020), en un problema complejo de optimización en el que no hay una teoría para encontrar o verificar lo óptimo, si varios algoritmos metaheurísticos con hiperparámetros afinados convergen en una solución, esta solución es muy probablemente la óptima.

A. Análisis de convergencia

Se evaluó la convergencia de las estimaciones de Q^2_{cum}, σ y h con respecto a la variación del número de iteraciones de cada algoritmo metaheurístico. En las Tablas 8 y 9 no se observa que el incremento de iteraciones haya afectado las medias y las desviaciones típicas de Q^2_{cum}, σ y h.

Se realizó el ANOVA tomando como hipótesis nula la igualdad de medias de Q^2_{cum}, σ y h para los diferentes número de iteraciones $g = \{10,30,50,100,150\}$. También se realizó la prueba de Levene para verificar la homogeneidad de varianzas. En la tabla 14 se presentan los resultados.

Tabla 14. ANOVA y prueba de homogeneidad de varianzas. Q^2_{cum}, σ y h por número de iteraciones

	Conjunto de datos	ANOVA		Homogeneidad de varianzas	
		F	Sig.	Estad. Levene	Sig.
Q^2_{cum}	Enfermedad de Parkinson	1,335	0,256	3,839	0,004
	Rendimiento académico	0,405	0,805	1,842	0,119
σ	Enfermedad de Parkinson	1,659	0,158	0,844	0,497
	Rendimiento académico	2,037	0,088	2,389	0,050
h	Enfermedad de Parkinson*	-	-	-	-
	Rendimiento académico	1,320	0,261	3,100	0,015

**h* es un valor constante

A un nivel de significación α = 0,05, no hay evidencia estadística del efecto del aumento de las iteraciones en las estimaciones de Q^2_{cum}, σ y h realizada por los algoritmos metaheurísticos, en ambos conjuntos de datos. Sin embargo, para los datos de rendimiento académico y a un nivel de significación de α = 0,10 ya se rechaza la hipótesis nula de igualdad de medias en las estimaciones de σ con el aumento del número de iteraciones.

Un punto relevante es la comparación del costo computacional asociado a cada algoritmo. En las Tablas 8 y 9 se observa una relación directa entre el número de iteraciones y el tiempo de ejecución en minutos de los cuatro algoritmos metaheurísticos. GA y PSO tienen un mayor costo computacional asociado en ambos conjuntos de datos y para cualquier número de iteraciones. Sin embargo, los algoritmos NM y HJ tuvieron un tiempo de ejecución promedio notablemente reducido.

En el caso de los conjuntos de datos evaluados y esta tarea específica de optimización, los resultados no son concluyentes en lo que respecta a la contribución a la precisión de las estimaciones realizadas por el aumento de las iteraciones y/o el tiempo de ejecución de los algoritmos metaheurísticos. En las Figura 6 y 7 se muestran las estimaciones de Q^2_{cum} en ambos conjuntos de datos con respecto al tiempo de ejecución de los algoritmos. No se observa una relación directa entre las dos variables en ninguno de los dos casos.

Estos resultados son coherentes con los de otras publicaciones (García-Ródenas et al., 2020), que sugieren que los algoritmos determinísticos son más eficientes desde el punto de vista computacional que los algoritmos metaheurísticos inspirados en la naturaleza y se refieren a la lenta convergencia de estos últimos.

Debido a las diferencias en los objetivos y las metodologías, no fue posible comparar los resultados obtenidos con el conjunto de datos de medición de la voz en pacientes con la enfermedad de Parkinson. Los resultados obtenidos con el conjunto de datos de rendimiento académico verificaron las estimaciones obtenidas en la Sección 5.1.2. donde se utilizó exclusivamente GA como optimizador de la regresión KPLS.

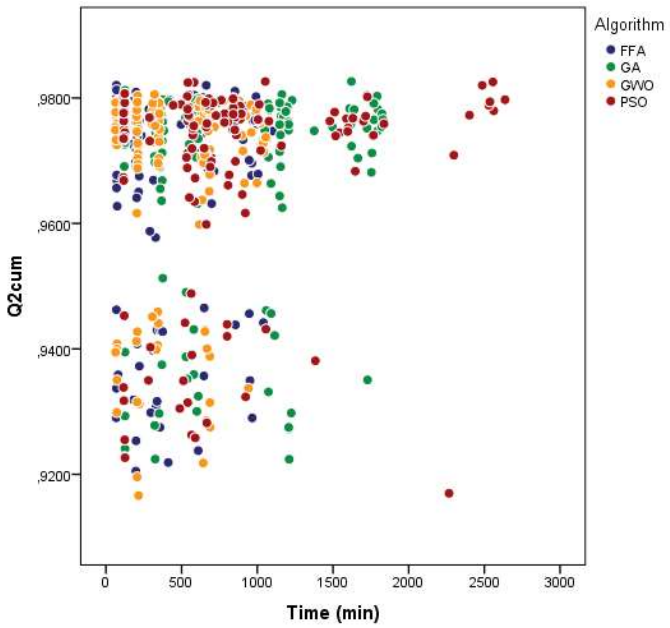


Figura 6. Estimaciones de Q^2_{cum} por tiempo de ejecución de los algoritmos metaheurísticos. Datos sobre la enfermedad de Parkinson

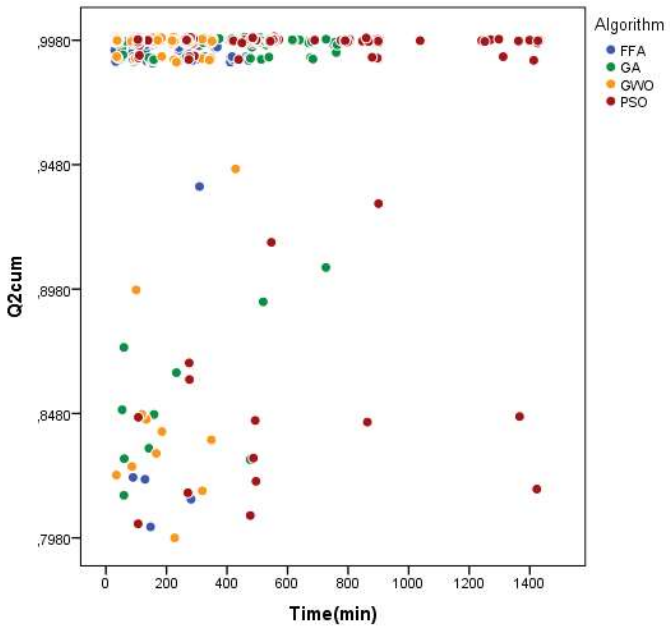


Figura 7. Estimaciones de Q^2_{cum} por tiempo de ejecución de los algoritmos metaheurísticos. Datos sobre rendimiento académico

5.3. Optimización con algoritmos meméticos

En esta sección se presenta un resumen de la metodología implementada y los resultados de la optimización de KPLS mediante MMA. Puede considerarse una extensión de las evaluaciones experimentales realizadas en la sección 5.2.; sin embargo, tiene un alcance más amplio y un análisis enfocado en la relación costo-beneficio en términos computacionales y de precisión de las estimaciones.

5.3.1. Metodología

La programación en el software R requirió la instalación los paquetes: *plsdepot*, *kernlab, metaheuristicOpt, Rmalschain* y *dfotimp* descritos en la Sección 4.3.1. Fue seleccionada la función kernel gaussiana (Ecuación 23). Se utilizaron los conjuntos de datos sobre enfermedad de Parkinson y rendimiento académico presentados en la Sección 4.2.2. Los hiperparámetros de los algoritmos metaheurísticos se detallan en la Tabla 7.

Se realizaron k = 30 ejecuciones el algoritmo de selección de parámetros de la regresión KPLS (Tabla 4). Se extrajo por cada ejecución: el valor máximo de Q^2_{cum} y sus valores asociados tanto del número de componentes h como del parámetro de la función kernel gaussiana σ para cada ejecución. Además, el tiempo de ejecución se calculó en minutos.

Se mantuvieron constantes el número de iteraciones en $g = 10$ y el tamaño de la población en p = 40. El análisis del desempeño de MMA como optimizador se realizó en referencia a los resultados alcanzados por el conjunto de algoritmos metaheurísticos: GA, PSO, FFA y GWO y los algoritmos determinísticos HJ y NM. Los indicadores evaluados son la media, la desviación estándar y el coeficiente de variación determinado por el cociente entre estos dos.

Se realizó un análisis estadístico para contrastar las hipótesis de diferencias entre las estimaciones y el tiempo de ejecución de los distintos algoritmos.

5.3.2. Resultados

Se realizaron k = 30 ejecuciones del algoritmo de selección de parámetros (Tabla 4) utilizando como optimizadores las metaheurísticas: MMA, GA, FFA, PSO y GWO y los algoritmos de búsqueda directa NM y HJ. Para generar la matriz kernel **K**, se utilizó la función

kernel gaussiana con parámetro σ > 0. El tamaño de la población de los algoritmos metaheurísticos GA, PSO, GWO y FFA se fijó en p = 40, y las iteraciones se aumentaron gradualmente por $g = 10$. Las características del ordenador donde se realizaron las pruebas son: Intel ® Celeron ® CPUG1610 @ 2.600GHz, 4.00 GB RAM.

En el algoritmo MMA se definió el tamaño de la población en p = 40 y número máximo de evaluaciones de la función objetivo `maxEvals` igual a 10. A pesar de las diferencias en la estructura algorítmica con relación a las demás metaheurísticas, este valor se toma en correspondencia al número de iteraciones. Los algoritmos NM y HJ tomaron 30 puntos de partida aleatorios $\sigma_0 > 0$ para ambos conjuntos de datos. El criterio de convergencia fue el valor $1e^{-06}$.

Las estimaciones promedio de Q^2_{cum}, σ y h fueron muy similares para todos los algoritmos evaluados. En ambos conjuntos de datos, MMA mostró una dispersión similar en las estimaciones del parámetro σ con respecto a los demás algoritmos metaheurísticos, pero un tiempo de ejecución muy inferior. Los detalles pueden verse en la Tabla 15.

Se ejecutó el ANOVA tomando como hipótesis nula la igualdad de medias de Q^2_{cum}, σ, h y tiempo de ejecución para los diferentes optimizadores. Los resultados presentados en la Tabla 16 indican que no existe evidencia estadística para rechazar esta hipótesis nula en cuanto a las estimaciones de Q^2_{cum}, σ, h en el conjunto de datos sobre la enfermedad de Parkinson. Sin embargo se rechaza la hipótesis nula para σ y h en el conjunto de datos sobre rendimiento académico.

Profundizando sobre este último resultado, se comprobó la no homogeneidad de varianzas a través de la prueba de Levene y se procedió a la ejecución de la prueba T2 Tamhane para verificar diferencias significativas entre las estimaciones de σ y h realizadas por MMA en comparación a los demás algoritmos. Los resultados se recogen en la Tabla 17.

Tabla 15. Media, desviación estándar y coeficiente de variación de las estimaciones de Q^2_{cum}, σ, h y tiempo de ejecución.

Algoritmo	Indicadores	Datos s/ rendimiento académico			Datos s/ enfermedad de Parkinson		
		Media	Desv. Est.	CV	Media	Desv. Est.	CV
MMA	Q^2_{cum}	0,9972	0,0015	0%	0,9656	0,0186	2%
	σ	3,9252	0,8118	21%	48,0542	4,12533	9%
	h	7	1	14%	6	0	0%
	Tiempo (min)	13,8995	0,5176	4%	16,5871	3,6008	22%
GA	Q^2_{cum}	0,9746	0,0557	6%	0,9709	0,0162	2%
	σ	3,8976	0,6541	17%	49,17002	3,31552	7%
	h	7	1	14%	6	0	0%
	Tiempo (min)	56,854	4,125	7%	124,2634	4,6336	4%
PSO	Q^2_{cum}	0,9857	0,0449	5%	0,9667	0,0196	2%
	σ	4,071	0,2088	5%	47,43298	3,48872	7%
	h	7	0	0%	6	0	0%
	Tiempo (min)	109,5227	6,0808	6%	119,6876	3,0775	3%
FFA	Q^2_{cum}	0,996	0,0027	0%	0,9687	0,0163	2%
	σ	3,8555	0,7062	18%	48,67281	2,79657	6%
	h	7	1	14%	6	0	0%
	Tiempo (min)	31,9841	1,7854	6%	70,6118	2,9303	4%
GWO	Q^2_{cum}	0,9918	0,0319	3%	0,97	0,0152	2%
	σ	4,1137	0,1283	3%	48,12832	3,33138	7%
	h	7	0	0%	6	0	0%
	Tiempo (min)	38,2993	2,6519	7%	71,103	3,8317	5%
HJ	Q^2_{cum}	0,9942	0,0039	0%	0,9568	0,0178	2%
	σ	5,8507	2,8899	49%	49,60078	20,94418	42%
	h	6	1	17%	6	0	0%
	Tiempo (min)	7,4195	0,8874	12%	18,0806	2,4198	13%
NM	Q^2_{cum}	0,9947	0,0038	0%	0,9652	0,0171	2%
	σ	6,7417	4,9002	73%	43,79257	10,64282	24%
	h	7	1	14%	6	0	0%
	Tiempo(min)	1,7819	0,3901	22%	5,1033	0,9318	18%

A partir de los resultados de las Tablas 16 y 17, se concluye que no se encuentra evidencia estadística sobre diferencias significativas entre estimaciones de $\mathrm{Q}^2_{\mathrm{cum}}, \sigma$ y h realizadas por MMA en comparación a los demás algoritmos.

Tabla 16. ANOVA de estimaciones Q^2_{cum}, σ, h y tiempo de ejecución por algoritmo.

ANOVA	Datos s/ rendimiento académico		Datos s/ enfermedad de Parkinson	
	F	Sig.	F	Sig.
$\mathrm{Q}^2_{\mathrm{cum}}$ vs. Algoritmo	1,708	0,122	2,104	0,055
σ vs. Algoritmo	7,889	0,000	1,023	0,412
h vs. Algoritmo	3,981	0,001	*	*
Tiempo vs. Algoritmo	3249,697	0,000	6037,745	0,000

*h es constante

Tabla 17. Prueba T2 Tamhane. Estimaciones de σ, h por MMA vs. otros algoritmos. Datos sobre rendimiento académico.

Algoritmos		σ			h		
(I)	(J)	Diferencia medias (I-J)	Error Est.	Sig.	Diferencia medias (I-J)	Error Est.	Sig.
MMA	FFA	0,069714	0,2049548	1	0,1	0,212	1
MMA	GA	0,027605	0,1970144	1	-0,04	0,201	1
MMA	GWO	-0,188506	0,1551858	0,996	0,25	0,162	0,939
MMA	PSO	-0,145752	0,1582338	1	0,18	0,168	0,999
MMA	HJ	-1,925473	0,6982161	0,234	0,88	0,351	0,338
MMA	NM	-2,816511	1,2744952	0,609	0,72	0,382	0,8

De especial importancia en esta sección es la relación costo-beneficio asociada a cada optimizador de la regresión KPLS. El costo computacional se define en términos del tiempo de ejecución en minutos mientras que el beneficio lo hace en términos de la dispersión en las estimaciones.

Se observa en la Tabla 15 que MMA tuvo un tiempo de ejecución considerablemente menor que los demás algoritmos metaheurísticos, y levemente mayor que los algoritmos de búsqueda directa. A fin de verificar estos resultados se ejecutó el ANOVA bajo la hipótesis nula

de igualdad de medias en el tiempo de ejecución para ambos conjuntos de datos. Los resultados de la Tabla 16 conllevan rechazar dicha hipótesis nula.

Seguidamente se verificó no homogeneidad de varianzas por la prueba de Levene, y se procedió a la ejecución de la prueba T2 Tamhane para verificar diferencias significativas en el tiempo de ejecución de MMA en comparación a los demás algoritmos. Los resultados se recogen en la Tabla 18.

Tabla 18. Prueba T2 Tamhane. Tiempo de ejecución en MMA vs. otros algoritmos

Algoritmos		Datos s/ rendimiento académico			Datos s/ enfermedad de Parkinson		
(I)	(J)	Diferencia medias (I-J)	Error Est.	Sig.	Diferencia medias (I-J)	Error Est.	Sig.
MMA	FFA	-18,084606	0,3572538	0,00	-54,024706	0,8475883	0,00
MMA	GA	-42,954506	0,7856725	0,00	-107,676261	1,0713779	0,00
MMA	GWO	-24,39981	0,4939482	0,00	-54,515878	0,9599969	0,00
MMA	PSO	-95,623177	1,1334058	0,00	-103,100457	0,8710759	0,00
MMA	HJ	6,480033	0,2309143	0,00	-1,49346	0,8008119	0,772
MMA	NM	12,117613	0,1404006	0,00	11,483822	0,6881322	0,00

La Tabla 18 muestra que MMA ha obtenido un tiempo de ejecución significativamente menor que los demás algoritmos metaheurísticos: GA, PSO, GWO y FFA, y poco distanciado de la velocidad de cómputo de los algoritmos determinísticos: HJ y NM, para ambos conjuntos de datos.

Capítulo 6

Conclusiones y prospectiva

6.1. Conclusiones

La tesis doctoral aborda el problema de optimizar la capacidad predictiva de la regresión KPLS y propone un procedimiento de ajuste metaheurístico para la selección de los parámetros de la función kernel θ y el número de componentes h.

Se plantea un enfoque de optimización diferente a las investigaciones actuales en el campo. Se define el problema de optimización tomando como función objetivo el coeficiente Q^2_{cum} y como argumentos de dicha función, los parámetros θ y h. El coeficiente Q^2_{cum} es un indicador de la capacidad predictiva global de la regresión KPLS que se obtiene a partir de procedimientos de validación cruzada. Toma valores en el intervalo entre 0 y 1, y cuanto mayor es su valor, indica un mejor rendimiento predictivo de la regresión.

Para resolver el problema, se diseña un algoritmo de selección de los parámetros θ y h que incorpora un optimizador iterativo general. Este optimizador puede ser un algoritmo metaheurístico basado en población, o con ajustes, un algoritmo determinístico libre de derivadas o de búsqueda directa. La implementación computacional con soporte del software R permite realizar diversas configuraciones del algoritmo de selección de parámetros; seleccionar funciones kernel y definir hiperparámetros de los optimizadores.

Se evalúa el desempeño como optimizadores: de algoritmos metaheurísticos inspirados en la naturaleza basados en población, tales como, algoritmos genéticos (GA), optimización de

enjambre de partículas (PSO), algoritmo de luciérnagas (FFA) y algoritmo basado en el comportamiento de lobos grises (GWO); algoritmos determinísticos libres de derivadas o de búsqueda directa como: Hooke-Jeeves (HJ) y Nelder-Mead (NM); y algoritmos meméticos (MMA), un algoritmo metaheurístico basado en población que integra búsqueda local. Las experimentaciones se realizan en diversos conjuntos de datos extraídos de repositorios digitales abiertos.

Los algoritmos evaluados como optimizadores han estimado parámetros óptimos (aproximados) de la regresión KPLS. No se observan diferencias significativas en las estimaciones promedio de Q^2_{cum}, θ, h para los diferentes algoritmos. Los algoritmos metaheurísticos inspirados en la naturaleza GA, PSO FFA y GWO demuestran una menor dispersión en la estimación de los parámetros θ, h. En contrapartida, requieren un mayor esfuerzo computacional que se incrementa significativamente con el aumento de iteraciones. Tampoco se obtuvieron resultados concluyentes sobre la convergencia de las estimaciones con el aumento de iteraciones.

Los algoritmos determinísticos HJ y NM son tomados en esta investigación con dos propósitos: validar los resultados obtenidos y establecer una referencia de la eficiencia de los algoritmos metaheurísticos. Se obtiene que HJ y NM obtienen estimaciones promedio similares a los algoritmos inspirados en la naturaleza, pero con una dispersión mayor en las estimaciones de los parámetros θ, h. De igual manera se constatan tiempos de ejecución considerablemente menores que los algoritmos metaheurísticos.

MMA ha demostrado un equilibrio entre la precisión de las estimaciones y el esfuerzo computacional que requiere. Las estimaciones Q^2_{cum}, θ, h han sido similares a las obtenidas por los demás optimizadores, con una menor dispersión en la estimación de los parámetros θ, h que los algoritmos determinísticos, y un tiempo de ejecución significativamente inferior al de los algoritmos metaheurísticos inspirados en la naturaleza.

Las conclusiones a las que se llegan en este documento son coherentes con los paradigmas actuales en regresión de mínimos cuadrados parciales y optimización metaheurística. Sin embargo, es necesario mencionar que las conclusiones se enmarcan al problema de

optimización planteado y están condicionadas a los conjuntos de datos evaluados y las herramientas computacionales escogidas.

6.2. Principales contribuciones

Los principales aportes de la tesis doctoral se corresponden a los objetivos trazados:

A. La determinación de una metodología de optimización de la capacidad predictiva de la regresión KPLS, la formulación matemática del problema y la correspondiente propuesta de solución, contemplada en el diseño de un algoritmo de ajuste metaheurístico de los parámetros de la regresión KPLS, constituyen los aportes centrales de este documento. La propuesta tiene características novedosas con respecto a otros abordajes que se encuentran en la literatura sobre el mismo problema, por lo que representa un aporte al estado del arte.
B. La exhaustiva experimentación con algoritmos de diferente naturaleza: metaheurísticos, de búsqueda directa y meméticos, ha permitido en primer lugar la validación del diseño del algoritmo, en segundo lugar, la comprobación mutua de los resultados y finalmente la identificación de los mejores solucionadores al problema de optimización planteado. Los resultados arrojados por los diferentes algoritmos son coherentes con los paradigmas actuales en materia de optimización metaheurística y numérica, y contribuyen al análisis y debate en ese campo.
C. El proceso de generación de herramientas computacionales para la optimización de la regresión KPLS inicia con las primeras experimentaciones, y a partir de allí, ya se encuentra en un proceso de mejora continua. El acoplamiento de paquetes y funciones del software R aquí escritos, permitirá a los usuarios de la técnica, encontrar un camino más rápido a una buena configuración de la regresión. Finalmente, la creación y documentación de un paquete capaz de automatizar la metodología propuesta es también una contribución a mencionar.

6.2.1. Publicaciones

Los resultados de esta tesis doctoral han sido publicados por la revista IEEE Access en el artículo "*KPLS Optimization with Nature-Inspired Metaheuristic Algorithms*". La revista IEEE

Access es una revista multidisciplinar con factor de impacto 3,745 según el Journal Citation Report (JCR) del año 2019 y ocupa el lugar 35 de 156 títulos (Q1) en la categoría "Computer Science and Information Systems".

Esta publicación es de especial relevancia porque presenta el algoritmo de selección de parámetros de la regresión KPLS propuesto en la tesis y comparte los principales resultados obtenidos. El documento es de acceso abierto y puede citarse de la siguiente manera:

> Mello-Román, J. D., & Hernández, A. (2020). KPLS Optimization with Nature-Inspired Metaheuristic Algorithms. *IEEE Access*, 8, 157482-157492. doi: 10.1109/ACCESS.2020.3019771.

La primera publicación en el marco de la tesis es una contribución al congreso 11th International Conference on Ambient Systems, Networks and Technologies (ANT) - International Workshop on Statistical Methods and Artificial Intelligence, denominada "*KPLS optimization approach using genetic algorithms*". La contribución fue sometida a un proceso de revisión de pares y aceptada para su presentación oral en Polonia en abril del 2020. El acta de la conferencia se halla publicada en Procedia Computer Science de ELSEVIER con acceso abierto. Esta publicación también reviste importancia por cuanto es la primera ocasión en que se comparte el enfoque de optimización propuesto en la tesis, y puede citarse como sigue:

> Mello-Román, J. D., & Hernandez, A. (2020). KPLS optimization approach using genetic algorithms. *Procedia Computer Science*, *170*, 1153-1160. doi: 10.1016/j.procs.2020.03.051

Otras publicaciones elaboradas y publicadas en la etapa de formación doctoral, y enmarcadas en la línea de investigación *Modelos Estadísticos* son las siguientes:

- Mello-Román, J. D., Mello-Román, J. C., Gomez-Guerrero, S., & García-Torres, M. (2019). Predictive Models for the Medical Diagnosis of Dengue: A Case Study in Paraguay. *Computational and mathematical methods in medicine*, *2019*. [JCR 1,77 – 2019 (Q3)]
- Mello Román, J. D., & Hernández Estrada, A. (2019). Un estudio sobre el rendimiento académico en Matemáticas. *Revista electrónica de investigación educativa*, *21*. [SJR 0,37

– 2019 (Q2)]

6.3. Futuros trabajos y líneas de investigación

Quedan varias preguntas abiertas que pueden guiar trabajos futuros:

La selección de los parámetros de la regresión KPLS es un tema que debe ser profundizado, tanto desde una perspectiva teórica como práctica. Son necesarios más estudios experimentales, para contrastar los resultados de esta tesis, con los de otros enfoques de optimización de la regresión KPLS.

Se prevé seguir implementando el algoritmo de selección de parámetros, en más conjuntos de datos y con una diversidad más amplia de optimizadores. Hasta el momento se han utilizado algoritmos metaheurísticos inspirados en la naturaleza, en su forma básica o tradicional. Es posible encontrar en la literatura, numerosas adaptaciones y mejoras a estos métodos, que también podrían explorarse.

Los algoritmos meméticos han demostrado un rendimiento satisfactorio en términos de precisión y velocidad de cómputo. La necesidad de determinar el mejor balance entre exploración y búsqueda local en estos algoritmos, es una consecuencia inmediata de los resultados de la tesis.

Una interrogante adicional es determinar cómo afecta la dimensión del conjunto de datos en la tarea de optimizar la regresión KPLS. Hasta el momento se han utilizado conjuntos de datos de tamaño razonable, que para algunos optimizadores han demandado un esfuerzo computacional importante.

Sigue siendo un problema abierto la selección de la función kernel en KPLS, al igual que en otras técnicas multivariadas que introducen el concepto de kernel cuando las estructuras de datos muestran comportamientos no lineales, como KPCA, KCCA (Correlación Canónica con funciones kernel) entre otros. Es una extensión natural de esta línea de investigación, la búsqueda de una configuración óptima de la función kernel para su integración a otras técnicas multivariadas.

Finalmente, continuarán los esfuerzos por consolidar una línea de investigación en técnicas multivariadas de modelización de datos con estructuras lineales y no lineales y realizar contribuciones significativas en este campo.

Referencias

[1] Abdi, H., (2010). Partial least squares regression and projection on latent structure regression (PLS Regression). *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(1), 97-106.

[2] Agrawal, V., & Sharma, A. (2010). Prediction of slump in concrete using artificial neural networks. *World Academy of Science, Engineering and Technology*, *45*, 25-32.

[3] Aquino, F. (2018). Optimización de cadenas de adición. *Universidad Tecnológica Nacional. Facultad Regional Concepción Del Uruguay.* (Tesis de Maestría). Recuperado de: http://ria.utn.edu.ar/handle/20.500.12272/3074

[4] Arranz de la Peña, J., & Parra Truyol, A. (2020). Algoritmos Genéticos. Recuperado de: http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf

[5] Asghari, S., & Navimipour, N. J. (2018). Nature inspired meta-heuristic algorithms for solving the service composition problem in the cloud environments. *International Journal of Communication Systems*, 31(12), e3708.

[6] Attia, K. A., Nassar, M. W., El-Zeiny, M. B., & Serag, A. (2017). Firefly algorithm versus genetic algorithm as powerful variable selection tools and their effect on different multivariate calibration models in spectroscopy: A comparative study. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, *170*, 117-123.

[7] Bache, K., & Lichman, M. (2013). UCI Machine Learning Repository [http://archive. ics. uci. edu/ml]. University of California, School of Information and Computer Science. *Irvine, CA*.

[8] Benmalek, E., Elmhamdi, J., & Jilbab, A. (2015). UPDRS tracking using linear regression and neural network for Parkinson's disease prediction. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, *4*(6), 189-193.

[9] Bennett, K. P., & Embrechts, M. J. (2003). An optimization perspective on kernel partial least squares regression. *Nato Science Series sub series III computer and systems sciences*, *190*, 227-250

[10] Bergmeir, C., Molina, D., & Benıtez, J. M. (2012). Rmalschains: Continuous Optimization using Memetic Algorithms with Local Search Chains (MA-LS-Chains) in R. *Journal of statistical software*.

[11] Binard, C. (2012). Introduction à la regression PLS (Groupe de travail PLS). Recuperado de: https://docplayer.fr/42840787-Introduction-a-la-regression-pls-carole-binard.html

[12] Bischl, B., Mersmann, O., Trautmann, H., & Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, *20*(2), 249-275.

[13] Brownlee, J. (2011). *Clever Algorithms: Nature-inspired Programming Recipes*. Jason Brownlee.

[14] Carrero Yubero, C. (2011). *Análisis de técnicas evolutivas para la estimación de curvas de tipos de interés* (Master's thesis).

[15] Chandwani, V., Agrawal, V., & Nagar, R. (2014). Modeling and analysis of concrete slump using hybrid artificial neural networks. *International Journal of Civil, structural, Construction and Architectural Engineering,(8)*, (9).

[16] Consonni, V., Ballabio, D., & Todeschini, R. (2010). Evaluation of model predictive ability by external validation techniques. *Journal of chemometrics*, *24*(3-4), 194-201.

[17] Cortez, P., & Silva, A. (2008). Using data mining to predict secondary school student performance. Recuperado de http://piano.dsi.uminho.pt/~pcortez/student.pdf

[18] de Almeida, V. E., de Araújo Gomes, A., de Sousa Fernandes, D. D., Goicoechea, H. C., Galvão, R. K. H., & Araújo, M. C. U. (2018). Vis-NIR spectrometric determination of Brix and sucrose in sugar production samples using kernel partial least squares with interval selection based on the successive projections algorithm. *Talanta*, *181*, 38-43.

[19] Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3-18.

[20] Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (Vol. 4, pp. 1942-1948).

[21] Fister, I., Fister Jr, I., Yang, X. S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, *13*, 34-46.

[22] Fiszelew, A. B. E. L., García Martínez, R., & de Buenos Aires, T. (2002). Generación automática de redes neuronales con ajuste de parámetros basado en algoritmos genéticos. *Revista del Instituto Tecnológico de Buenos Aires*, *26*, 76-101.

[23] Fogel, D. B. (2006). *Evolutionary computation: toward a new philosophy of machine intelligence* (Vol. 1). John Wiley & Sons.

[24] Frost, V. J., & Molt, K. (1998). Use of a genetic algorithm for factor selection in principal component regression. *Journal of Near Infrared Spectroscopy*, *6*(201), A185-A190.

[25] Fu, Y., Kruger, U., Li, Z., Xie, L., Thompson, J., Rooney, D. & Yang, H. (2017). Cross-validatory framework for optimal parameter estimation of KPCA and KPLS models. *Chemometrics and Intelligent Laboratory Systems*, *167*, 196-207.

[26] Gao, Y., Kong, X., Hu, C., Zhang, Z., Li, H., & Hou, L. A. (2015). Multivariate data modeling using modified kernel partial least squares. *Chemical Engineering Research and Design*, *94*, 466-474.

[27] Gao, Z. M., & Zhao, J. (2019). An improved grey wolf optimization algorithm with variable weights. *Computational Intelligence and Neuroscience*, *2019*.

[28] García-Ródenas, R., García-García, J. C., López-Fidalgo, J., Martín-Baos, J. Á., & Wong, W. K. (2020). A comparison of general-purpose optimization algorithms for finding optimal approximate experimental designs. *Computational Statistics & Data Analysis*, *144*, 106844.

[29] Gendreau, M., & Potvin, J. Y. (Eds.). (2010). *Handbook of metaheuristics* (Vol. 2, p. 9). New York: Springer.

[30] Geladi, P., & Kowalski, B. R. (1986), Partial least-squares regression: a tutorial. *Analytica chimica acta*, *185*,1-17.

[31] Gestal, M., Rivero, D., Rabuñal, J., Dorado, J., & Pazos, A. (2010). *Introducción a los Algoritmos Genéticos y la Programación Genética*. Universidade da Coruña, Servizo de Publicacións.

[32] Gogna, A., & Tayal, A. (2013). Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*, *25*(4), 503-526.

[33] Goldberg, D. E. Genetic algorithms in search, optimization, and machine learning (1989). *New York, Addison-Wesley.*

[34] Goodarzi, M., & dos Santos Coelho, L. (2014). Firefly as a novel swarm intelligence variable selection method in spectroscopy. *Analytica chimica acta*, *852*, 20-27.

[35] Merelo Guervós, J. (2009). Informática evolutiva: Algoritmos genéticos. Recuperado de: http://nuclear.fis.ucm.es/COMP-PHYS-13/GA/ie.pdf

[36] Harshman, R., Ladefoged, P., & Goldstein, L. (1977). Factor analysis of tongue shapes. *The Journal of the Acoustical Society of America*, *62*(3), 693-707.

[37] Hernández Torres R., Irizar Mesa, M., Llanes Santiago, O., Câmara, L, D, T,, da Silva Neto, A, J., & Zumalacárregui de Cárdenas, L. M. (2014). Comparación de diferentes algoritmos metaheurísticos en la estimación de parámetros del modelo relacional general

de cromatografía líquida en columna. *Ingeniare, Revista chilena de ingeniería*, *22*(1), 14-25,

[38] Hibbert, D. B. (1993). Genetic algorithms in chemistry. *Chemometrics and Intelligent Laboratory Systems*, *19*(3), 277-293.

[39] Holland, J. H. (1975). Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*, 439-444.

[40] Hooke, R., & Jeeves, T. A. (1961). "Direct Search" Solution of Numerical and Statistical Problems. *Journal of the ACM (JACM),* 8(2), 212-229.

[41] Huang, H., Chen, B., & Liu, C. (2015). Safety monitoring of a super-high dam using optimal kernel partial least squares. *Mathematical Problems in Engineering*, *2015*.

[42] Jalali-Heravi, M., & Kyani, A. (2007). Application of genetic algorithm-kernel partial least square as a novel nonlinear feature selection method: activity of carbonic anhydrase II inhibitors. *European journal of medicinal chemistry*, *42*(5), 649-659.

[43] Jia, R., Mao, Z., Wang, F. and He, D. (2015). Self-tuning final product quality control of batch processes using kernel latent variable model. *Chemical Engineering Research and Design*, *94*, pp,119-130.

[44] Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, *47*(260), 583-621.

[45] Kvasov, D. E., & Mukhametzhanov, M. S. (2018). Metaheuristic vs. deterministic global optimization algorithms: The univariate case. *Applied Mathematics and Computation*, *318*, 245-259.

[46] Koch, I. (2013). *Analysis of Multivariate and High-Dimensional Data* (Vol. 32). Cambridge University Press.

[47] Larson, J., Menickelly, M., & Wild, S. M. (2019). Derivative-free optimization methods. *Acta Numerica*, *28*, 287-404.

[48] Lee, S., & Lee, D. K. (2018). What is the proper way to apply the multiple comparison test?. *Korean journal of anesthesiology*, *71*(5), 353.

[49] Lin, S. W., Ying, K. C., Chen, S. C., and Lee, Z. J. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert systems with applications*, *35*(4), pp,1817-1824

[50] Little, M., McSharry, P., Hunter, E., Spielman, J., & Ramig, L. (2008). Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *Nature Precedings*, 1-1.

[51] Liu, S., Tang, J., & Yan, D. (2015, April). Multi-Kernel Partial Least Squares Regression based on Adaptive Genetic Algorithm. In *2015 International Conference on Automation, Mechanical Control and Computational Engineering*. Atlantis Press.

[52] Mello-Román, J. D., & Hernandez, A. (2020). KPLS optimization approach using genetic algorithms. *Procedia Computer Science*, *170*, 1153-1160.

[53] Mello-Román, J. D., & Hernandez, A. (2020). KPLS Optimization with Nature-Inspired Metaheuristic Algorithms. *IEEE Access*, 8, 157482-157492.

[54] Mirjalili S., Mirjalili M. & Lewis A. (2014), "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61.

[55] Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, *826*, 1989.

[56] Moscato, P., & Cotta, C. (2010). A modern introduction to memetic algorithms. In *Handbook of metaheuristics* (pp. 141-183). Springer, Boston, MA.

[57] Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308-313.

[58] Niu, P., Niu, S., & Chang, L. (2019). The defect of the Grey Wolf optimization algorithm and its verification method. *Knowledge-Based Systems*, *171*, 37-43.

[59] Noorizadeh, H., & Farmany, A. (2012). Determination of partitioning of drug molecules using immobilized liposome chromatography and chemometrics methods. *Drug testing and analysis*, 4(2), 151-157

[60] Noorizadeh, H., Farmany, A., & Noorizadeh, M. (2011). Application of GA-PLS and GA-KPLS calculations for the prediction of the retention indices of essential oils. *Química Nova*, *34*(8), 1398-1404.

[61] Noorizadeh, H., Sobhan Ardakani, S., Ahmadi, T., Mortazavi, S. S., & Noorizadeh, M. (2013). Application of genetic algorithm-kernel partial least square as a novel non-linear feature selection method: partitioning of drug molecules. *Drug testing and analysis*, *5*(2), 89-95.

[62] Osaba, E., Carballedo, R., Diaz, F., Onieva, E., Masegosa, A. D., & Perallos, A. (2018). Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems. *Neurocomputing*, *271*, 2-8.

[63] Pérez, R. A., & González-Farias, G. (2013). Partial least squares regression on symmetric positive-definite matrices. *Revista Colombiana de Estadística*, *36*(1), 177-192.

[64] Piotrowski, A. P., Napiorkowski, M. J., Napiorkowski, J. J., & Rowinski, P. M. (2017). Swarm intelligence and evolutionary algorithms: Performance versus speed. *Information Sciences*, *384*, 34-85.

[65] Ramsey, P. H., & Ramsey, P. P. (2008). Power of pairwise comparisons in the equal variance and unequal sample size case. *British Journal of Mathematical and Statistical Psychology*, *61*(1), 115-131.

[66] Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, *56*(3), 1247-1293.

[67] Rivas, J. A. M., & Cerrillo, S. F. J. (2014). Un algoritmo genético para selección de kernel en Análisis de Componentes Principales con Kernels. *Investigación Operacional*, *35*(2), pp,148-157.

[68] Riza, L. S., & Nugroho, E. P. (2018). MetaheuristicOpt: An R Package for Optimisation Based on Meta-Heuristics Algorithms. *Pertanika Journal of Science & Technology*, *26*(3).

[69] Roeva, O., & Fidanova, S. (2018). Comparison of different metaheuristic algorithms based on intercriteria analysis. *Journal of Computational and Applied Mathematics*, *340*, 615-628.

[70] Rosipal, R. & Trejo, L. J. (2001). Kernel partial least squares regression in reproducing kernel hilbert space. *Journal of machine learning research*, *2*(Dec), 97-123.

[71] Sakar, B. E., Isenkul, M. E., Sakar, C. O., Sertbas, A., Gurgen, F., Delil, S., ... & Kursun, O. (2013). Collection and analysis of a Parkinson speech dataset with multiple types of sound recordings. *IEEE Journal of Biomedical and Health Informatics*, *17*(4), 828-834.

[72] Sanchez, G. (2012). plsdepot: Partial least squares (PLS) data analysis methods. R package version 0.1, 17.

[73] Schölkopf, B., Smola, A., & Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, *10*(5), 1299-1319.

[74] Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

[75] Shingala, M. C., & Rajyaguru, A. (2015). Comparison of post hoc tests for unequal variance. *International Journal of New Technologies in Science and Engineering*, *2*(5), 22-33.

[76] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, *36*(2), 111-133.

[77] Sun, Y., Sahinidis, N. V., Sundaram, A. & Cheon, M.-S. (2020). "Derivative-free optimization for chemical product design," *Current Opinion in Chemical Engineering*, vol. 27, pp. 98—106.

[78] Talbi, E. G. (2009). *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons.

[79] Tan, P., Steinbach, M., & Kumar, V. (2006). *Introduction to data mining*. Pearson Education Limited.

[80] Tang, J., Zhang, J., Wu, Z., Liu, Z., Chai, T., & Yu, W. (2017). Modeling collinear data using double-layer GA-based selective ensemble kernel partial least squares algorithm. *Neurocomputing*, *219*, 248-262.

[81] Tenenhaus, M. (1998). *La régression PLS: théorie et pratique*, Editions technip.

[82] Thévenot, E. A. (2016). ropls: PCA, PLS (-DA) and OPLS (-DA) for multivariate analysis and feature selection of omics data.

[83] Tsanas, A., Little, M. A., McSharry, P. E., & Ramig, L. O. (2009). Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests. *IEEE transactions on Biomedical Engineering*, 57(4), 884-893.

[84] Shieber, S. M. (Ed.). (2004). *The Turing test: verbal behavior as the hallmark of intelligence*. Mit Press.

[85] Varadhan, R., & Borchers, H. W. (2016). dfoptim: Derivative-free optimization. R package version 2016.7-1.

[86] Vidal, P. J., & Olivera, A. C. (2018). DNA fragment assembling using a novel GPU firefly algorithm. *Dyna*, *85*(204), 108-116.

[87] Wang, Y., Huang, J. J., Zhou, N., Cao, D. S., Dong, J., & Li, H. X. (2015). Incorporating PLS model information into particle swarm optimization for descriptor selection in QSAR/QSPR. *Journal of Chemometrics*, *29*(12), 627-636.

[88] Wold, H. (1975). Soft modelling by latent variables: the non-linear iterative partial least squares (NIPALS) approach. *Journal of Applied Probability*, *12*(S1), 117-142.

[89] Wold, S., Sjostrom, M., & Eriksson, L. (2001). PLS-regression: a basic tool of chemometrics. *Chemometrics Intelligent Laboratory Systems*, 58 (2), 109-130.

[90] Xue, S., & Yan, X. (2017). A new kernel function of support vector regression combined with probability distribution and its application in chemometrics and the QSAR modeling. *Chemometrics and Intelligent Laboratory Systems*, *167*, 96-101.

[91] Yang, X. S. (2010). *Nature-inspired metaheuristic algorithms*. Luniverpress.

[92] Yeh, I. C. (2007). Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement and Concrete Composites*, *29*(6), 474-480.

[93] Yi, J., Huang, D., He, H., Zhou, W., Han, Q., & Li, T. (2017). A novel framework for fault diagnosis using kernel partial least squares based on an optimal preference matrix. *IEEE Transactions on Industrial Electronics*, *64*(5), 4315-4324.

[94] Ying, Y., Jin, W., Yu, H., Yu, B., Shan, J., Lv, S., ... & Mu, Y. (2017). Development of particle swarm optimization–support vector regression (PSO-SVR) coupled with microwave plasma torch–atomic emission spectrometry for quality control of ginsengs. *Journal of Chemometrics*, *31*(1), e2862.

[95] Zeileis, A., Hornik, K., Smola, A. & Karatzoglou, A. (2004). Kernlab - an S4 package for kernel methods in R. *Journal of statistical software*, *11*(9), 1-20.

[96] Zhang, H., Liu, S., Moraca, S., & Ojstersek, R. (2017). An effective use of hybrid metaheuristics algorithm for job shop scheduling problem. *International Journal of Simulation Modelling (IJSIMM)*, *16*(4).

[97] Žuvela, P., Macur, K., Liu, J. J., & Bączek, T. (2016). Exploiting non-linear relationships between retention time and molecular structure of peptides originating from proteomes and comparing three multivariate approaches. *Journal of pharmaceutical and biomedical analysis*, *127*, 94-100.

Anexos

En la sección de Anexos A.1. se presenta la codificación en el software R del algoritmo de selección de parámetros de la regresión KPLS, utilizada en las evaluaciones experimentales del Capítulo 5 y para el conjunto de datos sobre rendimiento académico. Para la ejecución de los códigos que se describen en esta sección es necesaria la instalación previa de los paquetes *plsdepot, kernlab, metaheuristicOpt, dfoptim* y *Rmalschains.*

En el Anexo A.2. se detalla la codificación del paquete *optimKPLS*. Su función `optim_kpls` retorna la estimación puntual del parámetro σ de la función kernel gaussiana, el número de componentes h y el valor de la función objetivo Q^2_{cum}. Se encuentra en una etapa inicial de desarrollo y toma como optimizador iterativo MMA.

En la sección de Anexos A.3. se presentan los ensayos realizados sobre los conjuntos de datos: "cornell" y concreto de alto rendimiento HPC. Las evaluaciones se realizan en la etapa de diseño del algoritmo de selección de parámetros de la regresión KPLS y ponen a prueba el mismo para varias funciones kernel.

A.1. Codificación en R para evaluaciones experimentales

A.1.1. Codificación con optimizador GA

```
numCompoments <- 10
dataset <-
read_excel("C:/Users/User/Documents/Proyecto01/Resultados_02/student_mat/stud
ent_mat.xlsx")
datosnuevos <- data.frame(dataset)
datos<-data.matrix(datosnuevos[1:43])
#FUNCION_OBJETIVO
funcion_objetivo <- function(x){
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  #vScale <- x[2]
  #if(vScale<=0){
    #  vScale <- 0.00001
    #}
  #vOffset <- x[3]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
```

```
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
    }
  }

  return(s[1])  # Q2cum -> retorna el valor maximo

}

##
funcion_objetivo_2 <- function(x){ #Para evaluacion final
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  #vScale <- x[2]
  #if(vScale<=0){
    #  vScale <- 0.00001
    #}
  #vOffset <- x[3]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
```

```
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
      s[3] <- vDegree
      #s[4] <- vScale
      #s[5] <- vOffset
    }
  }

  return(s)  # retorna vector de valores

}
## Define parameter
## Mejor hiperparametro
## iter 10,30,50,100,150
Pm <- 0.5
Pc <- 0.5
## Experimento 2
## Experimento 3
## Experimento 4

numVar <- 2
rangeVar <- matrix(c(0,10), nrow=2)
```

```
resultado <- matrix(0,30,7)

for (iter in 1:30) {
  tinicial <- proc.time() # Inicia el cronómetro
  ## calculate the optimum solution using Genetic Algorithm
  resultGA <- GA(funcion_objetivo, optimType="MAX", numVar, numPopulation=40,
                 maxIter=10, rangeVar, Pm, Pc) ##
  tfinal <- proc.time()-tinicial    # Detiene el cronómetro
  ## calculate the optimum value using sphere function
  optimum.value <- funcion_objetivo_2(resultGA)
  ##########################################################################
  #Guardar resultados
  ##########################################################################
  resultado[iter,1]<-iter             #iter
  resultado[iter,2]<-optimum.value[1] #Q2cum maximo funcion 2
  resultado[iter,3]<-optimum.value[2] #Num componente
  resultado[iter,4]<-optimum.value[3] #degree
  #resultado[iter,5]<-optimum.value[4] #scale
  #resultado[iter,6]<-optimum.value[5] #offset
  resultado[iter,7]<-tfinal[3]        #Tiempo de computo
  print(resultado)
}
Res <- data.frame(resultado)
write.csv2(Res,"C:/Users/User/Documents/Proyecto01/Resultados_02/student_mat/
GA_rbfdot_e2_40_10_05_05.csv")
```

A.1.2. Codificación con optimizador PSO

```
numCompoments <- 10
dataset <-
read_excel("C:/Users/User/Documents/Proyecto01/Revista_02/student_mat/student
_mat.xlsx")
datosnuevos <- data.frame(dataset)
datos<-data.matrix(datosnuevos[1:43])
#FUNCION_OBJETIVO
funcion_objetivo <- function(x){
```

```
vDegree <- x[1]
if(vDegree==0){
  vDegree <- 0.00001
}
vScale <- x[2]
if(vScale<=0){
  vScale <- 0.00001
}
vOffset <- x[3]
vOffset <- x[3]
kergauss<-rbfdot(sigma = vDegree)
K<-kernelMatrix(kergauss,datos)
jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
Q2 <- jd_kpls$Q2cum
Res <- data.frame(Q2)
v <- c() #Declaramos un vector vacio
q <- length(Res) #Cantidad de columnas

for (i in 1:numCompoments){
  for (j in 1:q){
    if(is.null(Q2[i,j])){
      Q2[i,j] <- 0
    }else {
      Q2[i,j] <- Q2[i,j]
    }
  }
}

for (i in 1:numCompoments){
  v[i] <- Q2[i,q]
}

s <- c() #Declaramos un vector vacio
s[1] <- max(v)
for (i in 1:numCompoments){
  if(s[1]==v[i]){
    s[2] <- i
  }
```

```
  }

  return(s[1])  # Q2cum -> retorna el valor maximo

}

##
funcion_objetivo_2 <- function(x){ #Para evaluacion final
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  vScale <- x[2]
  if(vScale<=0){
    vScale <- 0.00001
  }
  vOffset <- x[3]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }
```

```
  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
      s[3] <- vDegree
      s[4] <- vScale
      s[5] <- vOffset
    }
  }

  return(s)  # retorna vector de valores

}
## Define parameter
Vmax <- 4          ## 2,3,4,5
ci <- 2            ## 1.49445,2
cg <- 2            ## 1.49445,2
w <- 0.729         ## 0.729
numVar <- 3
rangeVar <- matrix(c(0,10), nrow=2)

resultado <- matrix(0,30,7)

for (iter in 1:30) {
  tinicial <- proc.time() # Inicia el cronómetro
  ## calculate the optimum solution using Particle Swarm Optimization
Algorithm
  resultPSO <- PSO(funcion_objetivo, optimType="MAX", numVar,
numPopulation=40, maxIter=10, rangeVar, Vmax, ci, cg, w)
  tfinal <- proc.time()-tinicial    # Detiene el cronómetro
  ## calculate the optimum value using sphere function
  optimum.value <- funcion_objetivo_2(resultPSO)
  ########################################################################
  #Guardar resultados
  ########################################################################
  resultado[iter,1]<-iter             #iter
```

```
  resultado[iter,2]<-optimum.value[1] #Q2cum maximo funcion 2
  resultado[iter,3]<-optimum.value[2] #Num componente
  resultado[iter,4]<-optimum.value[3] #degree
  resultado[iter,5]<-optimum.value[4] #scale
  resultado[iter,6]<-optimum.value[5] #offset
  resultado[iter,7]<-tfinal[3]        #Tiempo de computo
}
Res <- data.frame(resultado)
write.csv2(Res,"C:/Users/User/Documents/Proyecto01/Revista_02/student_mat/PSO
_rbfdot_e2_40_10_vmax_4_2.csv")
```

A.1.3. Codificación con optimizador FFA

```
numCompoments <- 10
dataset <-
read_excel("C:/Users/User/Documents/Proyecto01/Revista_02/student_mat/student
_mat.xlsx")
datosnuevos <- data.frame(dataset)
datos<-data.matrix(datosnuevos[1:43])
#FUNCION_OBJETIVO
funcion_objetivo <- function(x){
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  #vScale <- x[2]
  #if(vScale<=0){
  #  vScale <- 0.00001
  #}
  #vOffset <- x[3]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas
```

```
  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
    }
  }

  return(s[1])  # Q2cum -> retorna el valor maximo

}

##
funcion_objetivo_2 <- function(x){ #Para evaluacion final
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  #vScale <- x[2]
  #if(vScale<=0){
  #vScale <- 0.00001
  #}
  #vOffset <- x[3]
```

```
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
      s[3] <- vDegree
      #s[4] <- vScale
      #s[5] <- vOffset
    }
  }

  return(s)  # retorna vector de valores

}
## Define parameter
B0 <- 1       ## 0.6
```

```
gamma <- 1    ## 0.6
alpha <- 0.5 ## 0.5
numVar <- 2
rangeVar <- matrix(c(0,10), nrow=2)

resultado <- matrix(0,30,7)

for (iter in 1:30) {
  tinicial <- proc.time() # Inicia el cronómetro
  ## calculate the optimum solution using Firefly Algorithm
  resultFFA <- FFA(funcion_objetivo, optimType="MAX", numVar,
numPopulation=40,maxIter=10, rangeVar, B0, gamma, alpha)
  tfinal <- proc.time()-tinicial    # Detiene el cronómetro
  ## calculate the optimum value using funcion_objetivo function
  optimum.value <- funcion_objetivo_2(resultFFA)

  ########################################################################
  #Guardar resultados
  ########################################################################
  resultado[iter,1]<-iter             #iter
  resultado[iter,2]<-optimum.value[1] #Q2cum maximo funcion 2
  resultado[iter,3]<-optimum.value[2] #Num componente
  resultado[iter,4]<-optimum.value[3] #degree
  #resultado[iter,5]<-optimum.value[4] #scale
  #resultado[iter,6]<-optimum.value[5] #offset
  resultado[iter,7]<-tfinal[3]        #Tiempo de computo
  print(resultado)
}
Res <- data.frame(resultado)
write.csv2(Res,"C:/Users/User/Documents/Proyecto01/Revista_02/student_mat/FFA
_rbfdot_e2_40_10_1_1_05.csv")
```

A.1.4. Codificación con optimizador GWO

```
numCompoments <- 10
```

```
dataset <-
read_excel("C:/Users/User/Documents/Proyecto01/Resultados_02/student_mat/stud
ent_mat.xlsx")
datosnuevos <- data.frame(dataset)
datos<-data.matrix(datosnuevos[1:43])
#FUNCION_OBJETIVO
funcion_objetivo <- function(x){
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  #vScale <- x[2]
  #if(vScale<=0){
  #  vScale <- 0.00001
  #}
  #vOffset <- x[3]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }
```

```
  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
    }
  }

  return(s[1])  # Q2cum -> retorna el valor maximo

}

##
funcion_objetivo_2 <- function(x){ #Para evaluacion final
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  #vScale <- x[2]
  #if(vScale<=0){
  #  vScale <- 0.00001
  #}
  #vOffset <- x[3]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
```

```
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
      s[3] <- vDegree
      #s[4] <- vScale
      #s[5] <- vOffset
    }
  }

  return(s)  # retorna vector de valores

}
## Define parameter
numVar <- 2
rangeVar <- matrix(c(0,10), nrow=2)

resultado <- matrix(0,30,7)

for (iter in 1:30) {
  tinicial <- proc.time() # Inicia el cronómetro
  ## calculate the optimum solution using grey wolf optimizer
  resultGWO <- GWO(funcion_objetivo, optimType="MAX", numVar,
numPopulation=40,maxIter=10, rangeVar)

  tfinal <- proc.time()-tinicial    # Detiene el cronómetro
  ## calculate the optimum value using funcion_objetivo function
  optimum.value <- funcion_objetivo_2(resultGWO)

  ###########################################################################
```

```
  #Guardar resultados
  ##############################################################################
  resultado[iter,1]<-iter             #iter
  resultado[iter,2]<-optimum.value[1] #Q2cum maximo funcion 2
  resultado[iter,3]<-optimum.value[2] #Num componente
  resultado[iter,4]<-optimum.value[3] #degree
  #resultado[iter,5]<-optimum.value[4] #scale
  #resultado[iter,6]<-optimum.value[5] #offset
  resultado[iter,7]<-tfinal[3]        #Tiempo de computo
  print(resultado)
}
Res <- data.frame(resultado)
write.csv2(Res,"C:/Users/User/Documents/Proyecto01/Resultados_02/student_mat/
GWO_rbfdot_e2_40_10.csv")
```

A.1.5. Codificación con optimizador HJ

```
numCompoments <- 10
dataset <-
read_excel("C:/Users/User/Documents/Proyecto01/Resultados_02/student_mat/stud
ent_mat.xlsx ")
datosnuevos <- data.frame(dataset)
datos<-data.matrix(datosnuevos[1:43])
#FUNCION_OBJETIVO
funcion_objetivo <- function(x){
  vDegree <- x[1]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
```

```
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
    }
  }

  return(s[1])  # Q2cum -> retorna el valor maximo

}

##
funcion_objetivo_2 <- function(x){ #Para evaluacion final
  vDegree <- x[1]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
```

```
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
      s[3] <- vDegree
    }
  }

  return(s)  # retorna vector de valores

}

resultado <- matrix(0,30,7)
c <- 1
iter <-1.116
limS <- 100
while (iter <limS) {

tinicial <- proc.time() # Inicia el cronómetro
par0 <- c(iter,0.0)
res<-hjk(par0, funcion_objetivo, control = list(maximize=TRUE))
print(res)
#tol= 0.01
tfinal <- proc.time()-tinicial    # Detiene el cronómetro
R <- funcion_objetivo_2(res$par[1])
#########################################################################
#Guardar resultados
```

```
#########################################################################
resultado[c,1]<-iter             #iter
resultado[c,2]<-R[1] #Q2cum maximo funcion 2
resultado[c,3]<-R[2] #Num componente
resultado[c,4]<-R[3] #degree
resultado[c,5]<-tfinal[3]        #Tiempo de computo
print('****************************************************')
print('****************************************************')
print(iter)
print('****************************************************')
print(resultado)
print('****************************************************')
iter = iter + 3.145
c = c + 1
}
Res <- data.frame(resultado)
write.csv2(Res,"C:/Users/User/Documents/Proyecto01/Resultados/hjk_rbfdot.csv"
)
```

A.1.6. Codificación con optimizador NM

```
numCompoments <- 10
dataset <-
read_excel("C:/Users/User/Documents/Proyecto01/Resultados_02/student_mat/stud
ent_mat.xlsx ")
datosnuevos <- data.frame(dataset)
datos<-data.matrix(datosnuevos[1:43])
#FUNCION_OBJETIVO
funcion_objetivo <- function(x){
  vDegree <- x[1]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas
```

```
  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
    }
  }

  return(s[1])  # Q2cum -> retorna el valor maximo

}

##
funcion_objetivo_2 <- function(x){ #Para evaluacion final
  vDegree <- x[1]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas
```

```
  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
      s[3] <- vDegree
    }
  }

  return(s)  # retorna vector de valores

}

resultado <- matrix(0,30,7)
c <- 1
iter <-1.225
limS <- 100
while (iter <limS) {

tinicial <- proc.time() # Inicia el cronómetro
par0 <- c(iter,0.0)
res<-nmk(fn=funcion_objetivo,par0, control = list(maximize=TRUE))
print(res)
#tol= 0.01
```

```
tfinal <- proc.time()-tinicial    # Detiene el cronómetro
R <- funcion_objetivo_2(res$par[1])
#############################################################################
#Guardar resultados
#############################################################################
resultado[c,1]<-iter             #iter
resultado[c,2]<-R[1] #Q2cum maximo funcion 2
resultado[c,3]<-R[2] #Num componente
resultado[c,4]<-R[3] #degree
resultado[c,5]<-tfinal[3]        #Tiempo de computo
print('****************************************************')
print('****************************************************')
print(iter)
print('****************************************************')
print(resultado)
print('****************************************************')
iter = iter + 3.225
c = c + 1
}
Res <- data.frame(resultado)
write.csv2(Res,"C:/Users/User/Documents/Proyecto01/Resultados/nmk_rbfdot.csv"
)
```

A.1.7. Codificación con optimizador MMA

```
numCompoments <- 10
dataset <-
read_excel("C:/Users/User/Documents/Proyecto01/Resultados_03/student_mat/stud
ent_mat.xlsx")
datosnuevos <- data.frame(dataset)
datos<-data.matrix(datosnuevos[1:43])
#FUNCION_OBJETIVO
funcion_objetivo <- function(x){
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
```

```
  #vScale <- x[2]
  #if(vScale<=0){
  #  vScale <- 0.00001
  #}
  #vOffset <- x[3]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
      s[2] <- i
    }
  }

  return(s[1])  # Q2cum -> retorna el valor maximo

}
```

```
##
funcion_objetivo_2 <- function(x){ #Para evaluacion final
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  #vScale <- x[2]
  #if(vScale<=0){
  #vScale <- 0.00001
  #}
  #vOffset <- x[3]
  kergauss<-rbfdot(sigma = vDegree)
  K<-kernelMatrix(kergauss,datos)
  jd_kpls = plsreg2(K[,],datosnuevos[,44:46],comps = numCompoments)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numCompoments){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numCompoments){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numCompoments){
    if(s[1]==v[i]){
```

```
      s[2] <- i
      s[3] <- vDegree
      #s[4] <- vScale
      #s[5] <- vOffset
    }
  }

  return(s)  # retorna vector de valores

}

resultado <- matrix(0,30,7)

for (iter in 1:30) {
  tinicial <- proc.time() # Inicia el cronómetro
  ## calculate the optimum solution using Memetic Algorithm
  res.fun <- malschains(function(x) {-funcion_objetivo(x)}, lower=c(0),
upper=c(100), maxEvals=10,
                        control=malschains.control(popsize=40, istep=100,
ls="cmaes", optimum=-5))
  print(res.fun$sol)
  print(res.fun$fitness)
  result <- res.fun$sol
  tfinal <- proc.time()-tinicial    # Detiene el cronómetro
  ## calculate the optimum value using funcion_objetivo function
  optimum.value <- funcion_objetivo_2(result)

  ##############################################################################
  #Guardar resultados
  ##############################################################################
  resultado[iter,1]<-iter             #iter
  resultado[iter,2]<-optimum.value[1] #Q2cum maximo funcion 2
  resultado[iter,3]<-optimum.value[2] #Num componente
  resultado[iter,4]<-optimum.value[3] #degree
  #resultado[iter,5]<-optimum.value[4] #scale
  #resultado[iter,6]<-optimum.value[5] #offset
  resultado[iter,7]<-tfinal[3]        #Tiempo de computo
  print(resultado)
```

```
}
Res <- data.frame(resultado)
write.csv2(Res,"C:/Users/User/Documents/Proyecto01/Resultados_03/student_mat/
MA_rbfdot_e2_40_10.csv")
```

A.2. Codificación en paquete optimKPLS

```
#' @title Optimizacion de KPLS mediante un algoritmmo memetico
#' @description Ajuste metaheurístico de parámetros de la regresión KPLS
#' @param datos1 conjunto de datos de variables independientes.
#' @param datos2 conjunto de datos de variables dependientes.
#' @param numC numero de componentes
#' @param Li cota inferior
#' @param Ls cota superior
#' @param Pob numero de poblacion
#' @param maxE numero de evaluacion maxima
#' @return sigma y numero de componente
#' @examples
#' numCompoments <- 7
#' path<-"parkinson.xlsx"
#' dataset <- read_excel(path)
#' database <- data.frame(dataset)
#' data1<-database[2:20]
#' data2<-database[,21:22]
#' lo<-0
#' up<-100
#' po<-40
#' mE<-100
#' Resultado<-optim_kpls(data1,data2,numCompoments,lo,up,po,mE)
#' print(Resultado)
#' @export
optimkpls <- function(datos1,datos2,numC,Li,Ls,Pob,maxE){

  resultado <- matrix(0,1,4)

  tinicial <- proc.time() # Inicia el cronómetro
  ## Calculo de la solucion optima por medio utilizando el algoritmo memetico
  res.fun <- malschains(function(x) {-
funcion_objetivo(x,datos1,datos2,numC)}, lower=c(Li), upper=c(Ls),
maxEvals=maxE,
                        control=malschains.control(popsize=Pob, istep=100,
ls="cmaes", optimum=-5))
  result <- res.fun$sol

  tfinal <- proc.time()-tinicial    # Detiene el cronómetro
  ##
  optimum.value <- funcion_objetivo_2(result,datos1,datos2,numC)

  ########################################################################
  #Guardar resultados
  ########################################################################
  resultado[1,1]<-optimum.value[1] #Q2cum maximo funcion 2
  resultado[1,2]<-optimum.value[2] #Num componente
  resultado[1,3]<-optimum.value[3] #degree
```

```
  resultado[1,4]<-tfinal[3]        #Tiempo de computo
  Res <- data.frame(resultado)
  return(Res)
}

#' Funcion Objetivo
#' @param x valor de sigma.
#' @param datos1 conjunto de datos de variables independientes.
#' @param datos2 conjunto de datos de variables dependientes.
#' @param numC numero de componentes.
#' @export
funcion_objetivo <- function(x,datos1,datos2,numC){
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  kergauss<-rbfdot(sigma = vDegree)
  datos1<-data.matrix(datos1)
  K<-kernelMatrix(kergauss,datos1)
  jd_kpls = plsreg2(K[,],datos2,comps = numC)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numC){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numC){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numC){
    if(s[1]==v[i]){
      s[2] <- i
    }
  }

  return(s[1])  # Q2cum -> retorna el valor maximo
```

```
}

#' Funcion Objetivo de Evaluacion Final
#' @param x valor de sigma.
#' @param datos1 conjunto de datos de variables independientes.
#' @param datos2 conjunto de datos de variables dependientes.
#' @param numC numero de componentes.
#' @export
funcion_objetivo_2 <- function(x,datos1,datos2,numC){ #Para evaluacion final
  vDegree <- x[1]
  if(vDegree==0){
    vDegree <- 0.00001
  }
  kergauss<-rbfdot(sigma = vDegree)
  datos1<-data.matrix(datos1)
  K<-kernelMatrix(kergauss,datos1)
  jd_kpls = plsreg2(K[,],datos2,comps = numC)
  Q2 <- jd_kpls$Q2cum
  Res <- data.frame(Q2)
  v <- c() #Declaramos un vector vacio
  q <- length(Res) #Cantidad de columnas

  for (i in 1:numC){
    for (j in 1:q){
      if(is.null(Q2[i,j])){
        Q2[i,j] <- 0
      }else {
        Q2[i,j] <- Q2[i,j]
      }
    }
  }

  for (i in 1:numC){
    v[i] <- Q2[i,q]
  }

  s <- c() #Declaramos un vector vacio
  s[1] <- max(v)
  for (i in 1:numC){
    if(s[1]==v[i]){
      s[2] <- i
      s[3] <- vDegree
    }
  }

  return(s)  # retorna vector de valores

}
```

A.3. Ensayos con otros conjuntos de datos y funciones kernel.

A.3.1. Bases de datos cornell. Funciones kernel: polinomial, gaussiana y laplaciana.

Fueron seleccionadas tres funciones kernel (Zeileis et al., 2004):

- *Polinomial:* $k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{scale}(\mathrm{x}_i, \mathrm{x}_j) + \mathrm{offset})^{\mathrm{degree}}$, siendo degree el parámetro en evaluación y haciendo $\mathrm{scale} = 1$ y $\mathrm{offset} = 1$
- *Gaussiana:* $k(\mathrm{x}_i, \mathrm{x}_j) = \exp(-\sigma \left\| \mathrm{x}_i - \mathrm{x}_j \right\|^2)$ siendo σ el parámetro en evaluación,
- *Laplaciana:* $k(\mathrm{x}_i, \mathrm{x}_j) = \exp(-\sigma \left\| \mathrm{x}_i - \mathrm{x}_j \right\|)$ siendo σ el parámetro en evaluación,

La metodología implementada es la descrita en la Sección 5.1.1. Los parámetros de las funciones kernel evaluadas son configurados como individuos del optimizador GA. La dispersión en las estimaciones de los parámetros de las funciones kernel está determinada por la ecuación 22.

La Tabla 19 presenta los resultados para $p = 100$ y el incremento en el número de generaciones en $g = \{10, 20, 30, 40, 50\}$.

La Figura 8 muestra la variación de las estimaciones del parámetro degree de la función kernel polinomial para un tamaño fijo de población $p = 100$ y número de generaciones: $\boldsymbol{g} = 10$ y $\boldsymbol{g} = 50$, respectivamente.

La Tabla 20 presenta los resultados para $g = 10$ y el incremento en el tamaño de la población en $p = \{100, 250, 500, 750, 1000\}$.

La Figura 9 muestra la variación de las estimaciones del parámetro degree de la función kernel polinomial para un número fijo de generaciones $g = 10$ y tamaño de población: $p = 100$ y $p = 1000$, respectivamente.

Tabla 19. Ensayos con la base de datos cornell. Incremento del número de generaciones.

Kernel	Indicadores	Número de Generaciones $\boldsymbol{g}$				
		10	20	30	40	50
Polinomial	Max Q^2_{cum}	0,9674	0,9674	0,9674	0,9674	0,9674
	degree	6,9711	6,9679	6,9670	6,9647	6,9671
	h	3	3	3	3	3
	D_{degree}	7,74%	3,72%	2,87%	2,02%	1,57%
	Tiempo (seg.)	26,12	51,62	79,66	103,25	134,92
Gaussiana	Max Q^2_{cum}	0,9559	0,9559	0,9559	0,9559	0,9559
	σ	3,7555	3,7871	3,7809	3,7910	3,7808
	h	2	2	2	2	2
	D_σ	17,67%	10,09%	5,63%	5,47%	6,60%
	Tiempo (seg.)	26,31	52,98	76,68	105,10	130,48
Laplaciana	Max Q^2_{cum}	0,9525	0,9525	0,9525	0,9525	0,9525
	σ	-0,6975	-0,7104	-0,6993	-0,7138	-0,7007
	h	3	2	3	3	3
	D_σ	57,15%	48,07%	40,29%	43,22%	51,78%
	Tiempo (seg.)	27,96	58,49	80,66	107,26	147,13

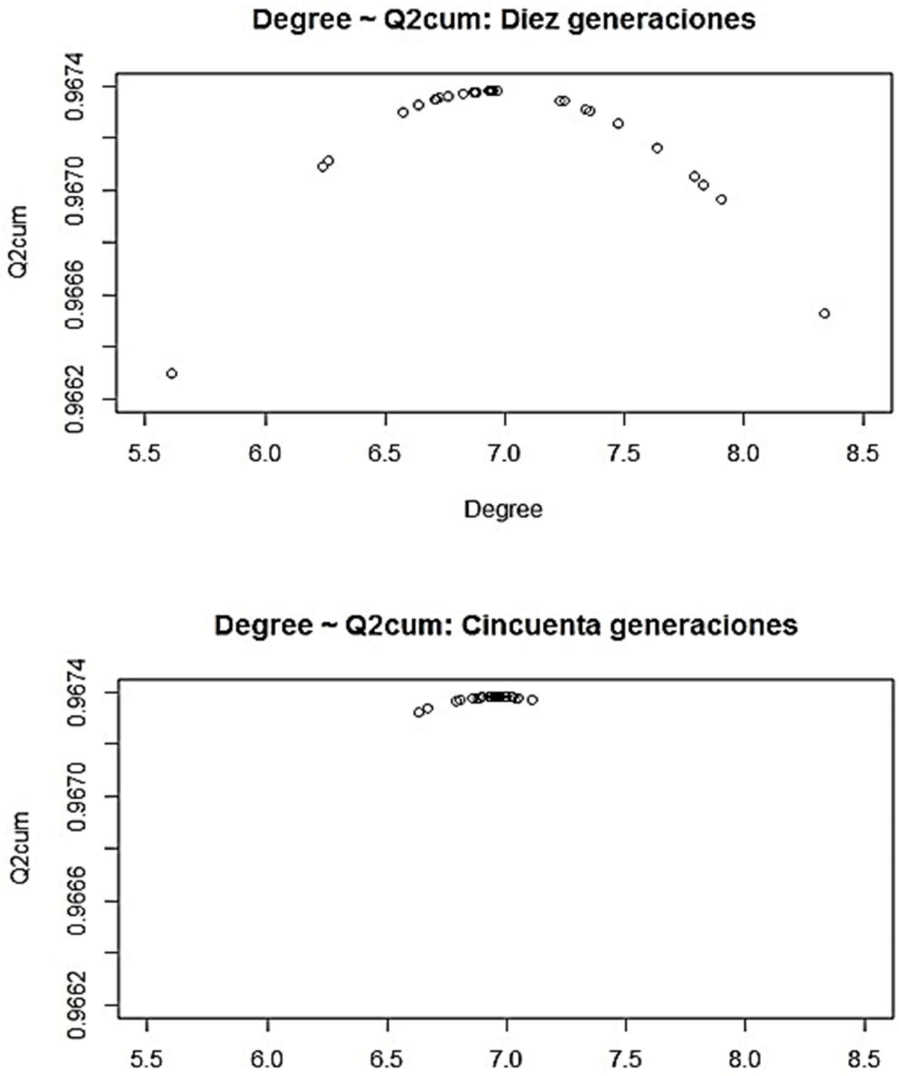


Figura 8. Ensayos con la base de datos cornell. Parámetro de la función kernel vs. Q^2_{cum} según número de generaciones

Tabla 20. Ensayos con la base de datos cornell. Incremento del tamaño de la población.

Kernel	Indicadores	Tamaño de población $\boldsymbol{p}$				
		100	250	500	750	1000
Polinomial	Max Q^2_{cum}	0,9674	0,9674	0,9674	0,9674	0,9674
	degree	6,9711	6,9642	6,9623	6,9649	6,9727
	h	3	3	3	3	3
	D_{degree}	7,74%	3,95%	1,30%	1,19%	0,71%
	Tiempo (seg.)	26,12	66,83	135,93	219,69	256,25
Gaussiana	Max Q^2_{cum}	0,9559	0,9559	0,9559	0,9559	0,9559
	σ	3,7555	3,7812	3,7814	3,7844	3,7812
	h	2	2	2	2	2
	D_σ	17,67%	7,00%	4,37%	2,42%	2,07%
	Tiempo (seg.)	26,31	68,66	147,71	210,86	287,75
Laplaciana	Max Q^2_{cum}	0,9525	0,9525	0,9525	0,9525	0,9525
	σ	-0,6975	-0,6881	-0,6998	-0,6969	-0,6951
	h	3	3	3	3	3
	D_σ	57,15%	30,63%	12,21%	17,60%	12,25%
	Tiempo (seg.)	27,96	70,98	141,77	221,68	347,43

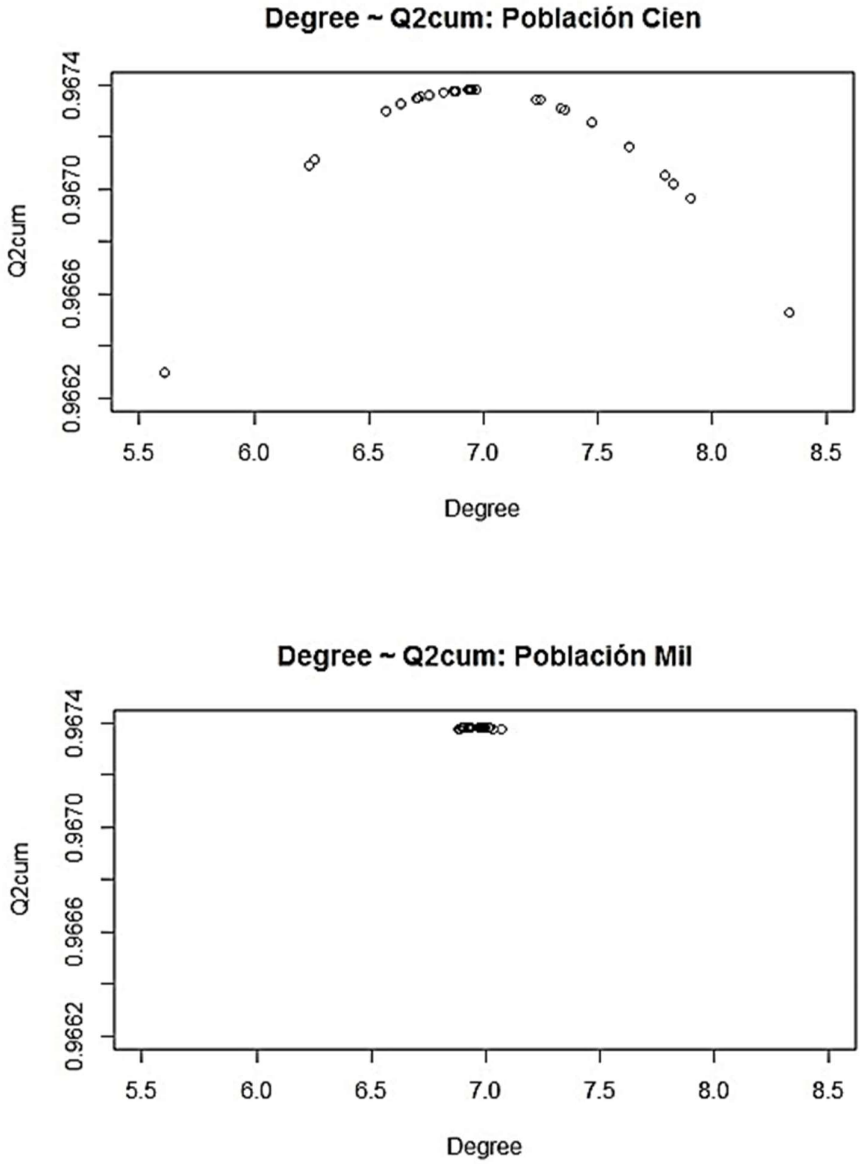


Figura 9. Ensayos con la base de datos cornell. Parámetro de la función kernel vs. Q^2_{cum} según tamaño de población

A.3.2. Bases de datos concreto de alto rendimiento (HPC). Función kernel polinomial.

Fue seleccionada la función kernel polinomial $k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{scale}(\mathrm{x}_i, \mathrm{x}_j) + \mathrm{offset})^{\mathrm{degree}}$ (Zeileis et al., 2004) donde $\theta = (degree,\ scale, offset)$ constituye un vector de parámetros de la función kernel polinomial, configurado como individuo para la tarea de optimización.

La metodología implementada es la descrita en la Sección 5.1.1. Los hiperparámetros de los optimizadores metaheurísticos PSO, FFA y GA (Yi et al., 2017; Attia et al., 2017) se presentan en la Tabla 21.

La media y el error estándar de las estimaciones de $\mathrm{Q}^2_{\mathrm{cum}}$, los parámetros $\theta = (degree,\ scale, offset)$, el tiempo de cómputo en segundos y el número óptimo de componentes *h* se presentan en la Tabla 22.

Tabla 21. Ensayos con la base de datos de concreto HPC. Hiperparámetros de optimizadores metaheurísticos

PSO	FFA	GA
$c_1 = 2$ aprendizaje individual	$\beta_0 = 1$ coeficiente de atracción	$p_c = 0.7$ probabilidad de cruce
$c_1 = 2$ aprendizaje colectivo	$\gamma = 1$ coeficiente de absorción	$p_m = 0.1$ probabilidad de
$v_{max} = 5$ velocidad máxima	$\alpha = 0.2$ parámetro aleatorización	mutación
$\omega = 0.7$ factor de inercia	$p = 100$ población	$p = 100$ población
$p = 100$ población	$g = 200$ iteraciones	$g = 200$ iteraciones
$g = 200$ iteraciones		

Tabla 22. Ensayos con la base de datos de concreto HPC. Estimaciones de $Q^2_{cum},\ \theta,\ h$ y tiempo de cómputo.

	PSO		FFA		GA	
	Media	Error estándar	Media	Error estándar	Media	Error estándar
Q^2_{cum}	0,5036	0,0030	0,4971	0,0036	0,5007	0,0036
$degree$	0,6114	0,2558	2,2857	0,2525	0,4921	0,2076
$scale$	2,3480	0,3850	5,0506	0,4086	5,6219	0,5176
$offset$	2,0988	0,3761	5,1957	0,5646	3,9798	0,6051
Tiempo (seg.)	4114,29	200,05	1352,58	57,18	2125,96	77,45
h	6	-	6	-	6	-