TESIS LICENCIATURA EN FÍSICA

CRITERIOS BÁSICOS PARA LA PRESENTACIÓN DE LA TESIS EN EL INSTITUTO BALSEIRO TANTO DE DOCTORADO COMO DE MAESTRÍA

J. Autor
Doctorando

Dr. J. Director
Director

Dr. J. Otro más
Co-director

Miembros del Jurado
Dr. J. J. Jurado (Instituto Balseiro)
Dr. Segundo Jurado (Universidad Nacional de Cuyo)
Dr. J. Otro Jurado (Univ. Nac. de LaCalle)
Dr. J. López Jurado (Univ. Nac. de Mar del Plata)
Dr. U. Amigo (Instituto Balseiro, Centro Atómico Bariloche)

21 de Enero de 2023

Colisiones Atómicas – Centro Atómico Bariloche

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

A mi familia
A mis amigos
A todos los que me conocen
A toda esa otra gente que no

Índice de símbolos

Índice de contenidos

Resumen

Este es el resumen en castellano.
La tesis debe reflejar el trabajo desarrollado, mostrando la metodología utilizada, los resultados obtenidos y las conclusiones que pueden inferirse de dichos resultados.

Palabras clave: FORMATO DE TESIS, LINEAMIENTOS DE ESCRITURA, INSTITUTO BALSEIRO

Abstract

This is the title in English:
The thesis must reflect the work of the student, including the chosen methodology, the results and the conclusions that those results allow us to draw.

Keywords: THESIS FORMAT, TEMPLATES, INSTITUTO BALSEIRO

Capítulo 1

Introducción

La ciencia de ondas gravitacionales es un área que vivió un rápido crecimiento en las últimas décadas. Este crecimiento llevó en septiembre del 2015 a la increíble primera detección obtenida de ondas gravitacionales producto de una colisión binaria de agujeros negros [1, 2]. Actualmente los interferometros LIGO, VIRGO y KAGRA van detectando cerca de 50 eventos, y planean empezar la cuarta serie de observaciones O4 a mediados de mayo de 2023 [1].

Cada observación no es solo una confirmación de una de las predicciones más relevantes de las ecuaciones de Einstein, sino que pueden aportan una gran cantidad de información de los fenómenos que las producen. De una onda producida por la colisión de dos agujeros negros se pueden inferir un total de 15 parámetros, de los cuales algunos son *intrínsecos* (como la relación entre las masas, o los espínes) y otros *extrínsecos* (la distancia del evento, la posición en el cielo, etc) [3].

La estimación de parámetros se realiza utilizando inferencia Bayesiana [4], un método que requiere generar funciones de onda para diversos parámetros en tiempo real. Esto es un problema debido a la dificultad que representa el resolver las ecuaciones de Einstein; para obtener la función de onda producto de una colisión binaria utilizando relatividad numérica se necesitan meses de cómputo. Debido a que se necesitan varias configuraciones al momento de realizar la inferencia de parámetros, la relatividad numérica no es una opción a la hora de generar las funciones de onda. En esta tésis se trabajará dentro del marco de los modelos sustitutos [5], una alternativa que logra generar resultados de alta precisión, comparable a aquella de los métodos de relatividad numérica, pero en tiempo real y en una simple computadora portátil.

La construcción de un modelo sustituto de orden reducido se puede explicar a grandes rasgos en tres etapas: 1) se seleccionan n configuraciones dentro de un dominio de parámetros para representar a todo el dominio, 2) se realiza una interpolación en el dominio temporal, 3) se vuelve a realizar una interpolación, pero esta vez en el espacio

[1] https://observing.docs.ligo.org/plan/

de los parámetros. Como resultado se obtiene un modelo predictivo capaz de generar resultados precisos y en tiempo real (este proceso está explicado en la siguiente review [6]).

Si bien el objetivo final es crear un modelo predictivo, este trabajo se centra solo en la primera parte de la construcción del modelo; la representación. Esto se logra utilizando bases reducidas [7–11] construidas a partir de un algoritmo voraz o *greedy*. Este método busca eliminar la redundancia que pueda estar presente en un conjunto de soluciones para un dado espacio de parámetros, de forma que se logre representar todo el espacio a partir de un número relativamente bajo de elementos.

Para ser más precisos, este trabajo se centra en la optimización de hiperparámetros para la construcción de una base reducida con refinamiento *hp-greedy* [14]. Este refinamiento divide el espacio de parámetros en distintos subdominios de forma iterativa, logrando una estructura de árbol binario. Una ventaja de esto es poder trabajar con espacios de parámetros que presenten discontinuidades, y otra es la reducción del tiempo de proyección de un espacio a la base reducida, lo que daría lugar a un modelo más rápido y eficiente. Una base reducida *hp-greedy* constituye un sistema de aprendizaje supervisado con un gran número de configuraciones de hiperparámetros, las cuales afectan en gran medida el rendimiento del sistema, por lo que resulta interesante concentrarse en el problema de optimizar la construcción de este sistema.

Un método de optimización cada vez más utilizado dentro de distintas áreas de la ciencia de datos es la optimización Bayesiana [23, 24], con resultados sobresalientes a la hora de entrenar distintos modelos de aprendizaje automático y aprendizaje profundo. Realizar una elección manual de los hiperparámetros de una base reducida *hp-greedy* lleva a resultados mediocres en general, por lo que es necesario realizar muchas pruebas antes de encontrar una mejora con respecto al resultado inicial. La optimización Bayesiana es una forma de automatizar este proceso a la vez que cada nueva evaluación será una elección informada a partir de las evaluaciones realizadas anteriormente. Por esto se optó por utilizar este método, obteniendo resultados bastante favorables en la optimización del error de representación de las bases construidas.

1.1. Representación Ondas Gravitacionales

ondas grav. rel num [12]

1.2. Datos utilizados

Se utilizaron ondas gravitacionales generadas a partir del modelo híbrido *NRHybSur3dq8*[13] de relativdad numérica y aproximaciones post Newtonianas para colisiones de agujeros negros binarios.

Cada onda h generada se representa por una serie temporal compleja de la forma:

$$h = h_+ + ih_\times$$

Recordando que h está parametrizada por λ

$$h = h(t;\lambda) = h_\lambda(t) = h_\lambda$$

En este caso λ tendrá 3 dimensiones, $(q, \chi_{1_Z}, \chi_{2_Z})$, y estará acotado de la siguiente manera:

- Relación entre masas q: $1 \leq q \leq 8$
- Espín del agujero negro más pesado (liviano) $\chi_{1_Z}(\chi_{1_Z})$: $|\chi_{1_Z}|, |\chi_{2_Z}| < 0{,}8$

Se representa un conjunto $\mathcal{K}$ de N muestras de λ de la siguiente forma:

$$\mathcal{K} = \{h_{\lambda_i}\}, \quad i = 1, ..., N$$

y debido a que cada h_{λ_i} es una serie temporal, se puede representar $\mathcal{K}$ en forma de la matriz $H \in \mathbb{C}^{N\times L}$:

$$H = \begin{bmatrix} h_{\lambda_1} \\ h_{\lambda_2} \\ \vdots \\ h_{\lambda_N} \end{bmatrix} = \begin{bmatrix} h_{\lambda_1}(t_1) & h_{\lambda_1}(t_2) & \cdots & h_{\lambda_1}(t_L) \\ h_{\lambda_2}(t_1) & h_{\lambda_2}(t_2) & \cdots & h_{\lambda_2}(t_L) \\ \vdots & \vdots & \ddots & \vdots \\ h_{\lambda_N}(t_1) & h_{\lambda_N}(t_2) & \cdots & h_{\lambda_N}(t_L) \end{bmatrix}$$

Siendo L la longitud de la serie temporal. De forma que cada fila de H es una onda gravitacional.

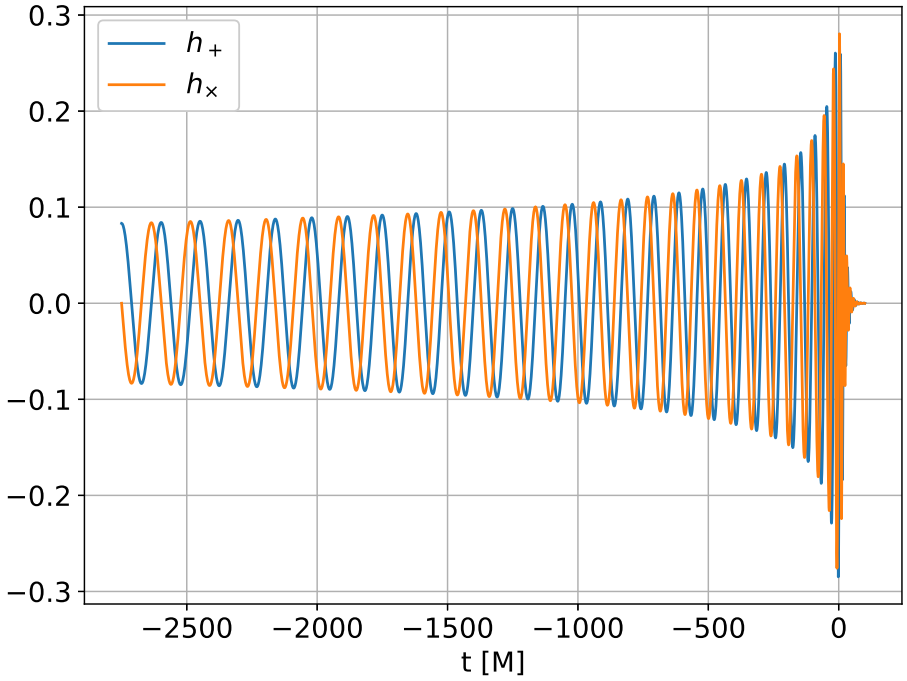

Figura 1.1: polarizaciones h_+ y $h_\times$ para $q = 3$, $\chi_{1z} = \chi_{2z} = 0$, en el modo $l = 2$, $m = 2$.

Capítulo 2

Teoría de Aproximaciones: Bases Reducidas y Aprendizaje

En la primera sección de este capítulo se dará una introducción a las bases reducidas, un método de aproximación fundamental en el modelado de orden reducido [6]. Este marco posee dos etapas bien diferenciadas; por un lado la parte de entrenamiento que puede ser bastante costosa (*offline*) pero que solo debe realizarse una vez, y después la parte de evaluación, que es mucho más rápida (*online*). Luego, en la segunda sección se ampliará con el refinamiento hp *greedy* [14]; una metodología que descompone de forma iterativa el dominio de parámetros para crear bases reducidas locales.

2.1. Bases Reducidas

El objetivo de este método es obtener una base que represente un espacio de soluciones de forma aproximada, evitando tener que resolver el problema completo múltiples veces para generar más soluciones. En este contexto las soluciones son ondas gravitacionales y resolver el problema consistiría en obtener soluciones a partir de relatividad numérica.

Las bases reducidas son un método de expansión espectral así como la expansión de Fourier o los ponimonios de Jacobi. La diferencia de este método es que los elementos de la base serán soluciones del espacio que se quiere aproximar (no necesariamente funciones trigonométricas o polinomios, como en otros métodos).

Partiendo de un espacio de soluciones $\mathcal{F} := \{h_\lambda = h_\lambda(t) = h(\lambda, t)\}$, donde h_λ representa una onda gravitacional parametrizada por λ, que generalmente es multidimensional y puede representar la relación entre las masas de los agujeros negros o los espínes, por ejemplo.

Para construir la base se parte de un conjunto de entrenamiento $\mathcal{T} := \{\lambda_i, h_{\lambda_i}\}_{i=1}^{N}$, resultado de un muestreo $\mathcal{K} := \{h_{\lambda_i}\}_{i=1}^{n}$de $\mathcal{F}$, y que contiene la información del conjunto

de parámetros $\{\lambda_i\}_{i=1}^n$ utilizado para la construcción de cada función de $\mathcal{K}$. A partir de $\mathcal{T}$ se construye una base $\{e_i\}_{i=1}^n$, generalmente con $n \ll N$. Luego se puede representar $\mathcal{K}$, e incluso $\mathcal{F}$ con la combinación lineal

$$h_\lambda(t) \approx \sum_{i=1}^{n} c_{i,\lambda} e_i(t). \tag{2.1}$$

2.1.1. Elección de una base óptima

La pregunta de que tan bien se puede aproximar el espacio $\mathcal{F}$ lleva a la distancia de Kolmogorov [15]:

$$d_n := \min_{\{e_i\}_{i=1}^n} \max_{\lambda\in\Omega} \min_{c_{i,\lambda}\in\mathbb{C}} ||h_\lambda - \sum_{i=1}^{n} c_{i,\lambda} e_i(t)||^2. \tag{2.2}$$

Básicamente es una medida del máximo error de representación en un espacio compacto de parámetros Ω, con una base y coeficientes óptimos. El producto interno $||\cdot||$ está dado por

$$||h_\lambda||^2 := \int_{t_i}^{t_f} |h_\lambda(t)|^2 dt$$

definido a partir del producto interno

$$\langle h_1, h_2 \rangle := \int_{t_i}^{t_f} \overline{h_1}(t) h_2(t) dt$$

Para entender la notación en d_n, analizando los mín, máx y mín de derecha a izquierda:

- $\min_{c_{i,\lambda}\in\mathbb{C}}$ se refiere a la representación óptima con respecto a la base utilizada. Esto implica realizar una proyección ortogonal a la base. Suponiendo una base ortonormal los coeficientes óptimos serán:

$$c_{i,\lambda} = \langle e_i, h_\lambda \rangle \tag{2.3}$$

Y se puede reescribir la ecuación (2.2) de la siguiente forma:

$$d_n := \min_{\{e_i\}_{i=1}^n} \max_{\lambda\in\Omega} ||h_\lambda - \mathcal{P}_n h_\lambda||^2.$$

con

$$\mathcal{P}_n h_\lambda = \sum_{i=1}^{n} c_{i,\lambda} e_i(t)$$

- $\max_{\lambda \in \Omega}$ hace referencia a que se tiene en cuenta el peor error (el máximo) en el espacio de parámetros Ω.
- Por último $\min_{\{e_i\}_{i=1}^n}$ implica elegir la base óptima que de lugar al *mejor peor error* en el dominio de parámetros.

Por lo tanto d_n sirve como un límite superior; es lo mejor que se puede lograr si la base fue escogida de forma óptima. En algunos casos d_n se puede calcular teóricamente [16]. Más específicamente se puede probar que $d_n \sim n^{-r}$ para funciones con sus primeras $(r-1)$ derivadas continuas y que $d_n \sim e^{-an^b}$ para funciones con una dependencia C^∞ [17]. En el contexto de ondas gravitacionales se espera un comportamiento asíntotico de convergencia exponencial con n [18, 19].

La elección de una base óptima es una tarea complicada de complejidad combinatoria y no aplicable en la práctica. Por lo tanto se utilizará un acercamiento más eficiente en base al uso de un algoritmo de tipo voraz o *greedy*, que da lugar a una base cuasi-óptima en un sentido matemático, de complejidad lineal. Para más detalles se recomienda ver [6].

2.1.2. Algoritmo *Greedy* para construir una base cuasi-óptima

Un algoritmo de tipo *greedy* es un proceso iterativo en el cual a cada paso se toma la elección que produzca el mayor beneficio inmediato. En el contexto de las bases reducidas, cada iteración del algoritmo agregará un elemento a la base, de forma que el elemento añadido será aquel que daba lugar al peor error de representación dentro del conjunto de entrenamiento $\mathcal{T}$ (de esta forma se reduce el peor error de representación). El error de entrenamiento de una base de dimensión n o "Error *greedy*", se define como

$$\sigma_n := \max_{\lambda \in \{\lambda_i\}_{i=1}^N} ||h_\lambda - \mathcal{P}_n h_\lambda||^2.$$

También se hablará de un *máximo error de validación*, que es el máximo error de representación que ocurre dentro de un conjunto de validación, generalmente más denso que el conjunto de entrenamiento.

El proceso de construcción de una base reducida se puede ver en el algoritmo 1, y se explica a continuación:

- Al algoritmo ingresan un conjunto de entrenamiento $\mathcal{T} = \{\lambda_i, h(\lambda_i)\}_{i=1}^N$ (que incluye tanto las funciones de onda como los parámetros de las funciones), el número máximo de elementos permitidos en la base n_{max}, y un mínimo error de tolerancia ε también llamado *tolerancia greedy*.
- El primer elemento de la base reducida se elije de forma arbitraria a partir del conjunto de parámetros $\{\lambda_i\}_{i=1}^N$. El primer parámetro elegido se llama *semilla* o

primer parámetro *greedy* Λ_1. Luego el primer elemento será $h(\Lambda_1)$, pero normalizado.

- Luego iterativamente se agregan nuevos elementos a la base, seleccionando los parámetros que den lugar al mayor error dentro del conjunto de entrenamiento. El conjunto de estos parámetros recibe el nombre de *parámetros greedy*.
- Por conveniencia a la hora de realizar proyecciones en el espacio generado por la base, se construye una base ortonormal (de forma que los coeficientes $c_{i,\lambda}$ se obtengan por medio de la ecuación (2.3)). Para esto se aplica un algoritmo de ortonormalización de Gram-Schmidt (líneas 9 y 10 del algoritmo 1).
- El algoritmo finaliza cuando se alcanzó el número máximo de elementos en la base $n = n_{max}$, o cuando el máximo error de representación σ_n sea menor al error de tolerancia ε, lo que ocurra primero. El algoritmo devuelve los elementos de la base reducida $\{e_i\}_{i=1}^n$, el conjunto de parámetros *greedy* $\{\Lambda_i\}_{i=1}^n$ y el error de representación σ_n.

Algoritmo 1 `GreedyRB`$(\mathcal{T}, \varepsilon, n_{max})$

Input: $\mathcal{T} = \{\lambda_i, h(\lambda_i)\}_{i=1}^N, \varepsilon, n_{max}$

1: $i = 1$
2: $\sigma_1 = 1$ ▷ Inicializar error de representación a 1
3: $\Lambda_1 = \lambda_k$ ▷ Elección arbitraria de la semilla
4: $e_1 = h(\Lambda_1)/||h(\Lambda_1)||$
5: $\text{rb} = \{e_1\}$ ▷ Primer elemento de la base reducida
6: **while** $\sigma_i > \varepsilon$ **and** $i < n_{max}$ **do**
7: $\quad i = i + 1$
8: $\quad \Lambda_i = arg\,\text{máx}_{\lambda\in\mathcal{T}} ||h_\lambda - \mathcal{P}_{i-1}h_\lambda||^2$ ▷ Selección del parámetro *greedy*
9: $\quad e_i = h(\Lambda_i) - P_{i-1}h(\Lambda_i)$ ▷ Gram-Schmidt
10: $\quad e_i = e_i/||e_i||$ ▷ Normalización
11: $\quad \text{rb} = \text{rb} \cup \{e_i\}$
12: $\quad \sigma_i = \text{máx}_{\lambda\in\mathcal{T}} ||h_\lambda - \mathcal{P}_i h_\lambda||^2$ ▷ Error de representación
13: **end while**

Output: $\text{rb} = \{e_i\}_{i=1}^n, \Lambda = \{\Lambda_i\}_{i=1}^n, \sigma_n$

2.1.3. Convergencia del Algoritmo *Greedy*

Para una tolerancia *greedy* ε el algoritmo *greedy* entrega una base reducida con un error

$$\sigma_n = \max_{\lambda\in\{\lambda_i\}_{i=1}^N} ||h_\lambda - \mathcal{P}_n h_\lambda||^2 \leq \varepsilon$$

Dado que la elección del primer elemento de la base reducida es arbitrario (la elección de la semilla), el lector o lectora podría preguntarse que tan relevante es esta elección a la hora de condicionar la convergencia del error de representación σ_n. Como se mencionó anteriormente, la medida de Kolmogorov d_n puede utilizarse como cota superior. Resultados obtenidos en [20] muestran que si d_n decae exponencialmente, σ_n también lo hará:

$$d_n \leq De^{-an^\alpha} \quad \rightarrow \quad \sigma_n \leq \sqrt{2D}\gamma^{-1}e^{-a'_\alpha n^\alpha},$$

con D, α, a constantes positivas y $\gamma \in (0, 1]$. De forma similar, si el decaimiento de d_n es polinomial, también lo será el decaimiento de σ_n:

$$d_n \leq Dn^{-\alpha} \quad \rightarrow \quad \sigma_n \leq D'_\alpha n^{-\alpha},$$

con D, $\alpha > 0$ y $\gamma \in (0, 1]$.

En la figura 2.1 se puede ver el decaimiento exponencial del error de representación en función del número de elementos de la base reducida aplicado en el contexto de ondas gravitacionales [21]. Lo más relevante es que se puee ver que este comportamiento no depende de la elección de la raíz; el área sombreada representa los extremos de todas las posibles semillas dentro del conjunto de entrenamiento.

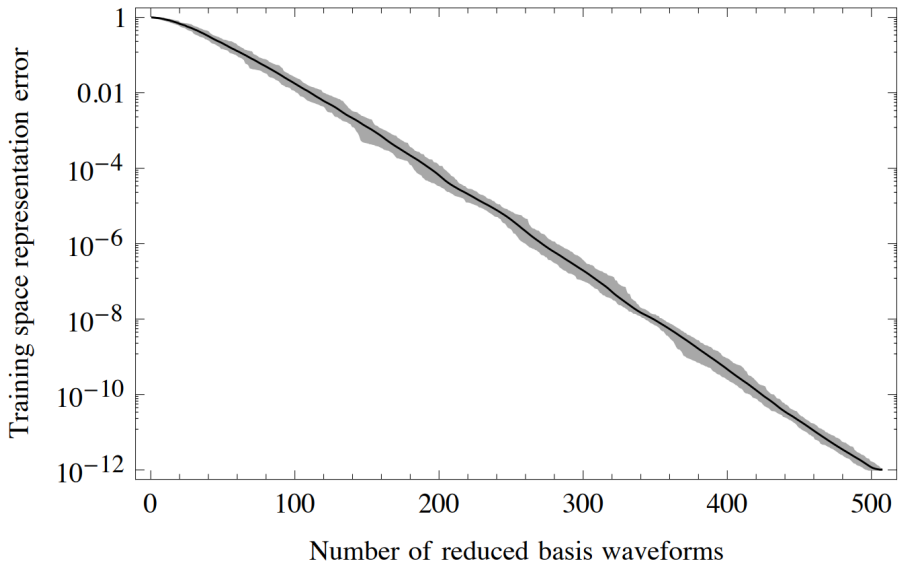

Figura 2.1: Error de representación en función del número de funciones de onda en la base reducida para un modo cuasinormal (QNM). Se probaron todas las semillas posibles del conjunto de entrenamiento; en negrita se ve el promedio, y el área sombreada representa los valores extremos [21].

2.2. Bases Reducidas hp Greedy

El nombre del método hp greedy [14] viene de la combinación del *"refinamiento p"* y del *"refinamiento h"*. El *refinamiento p* proviene de los métodos espectrales con bases polinomiales [22] y se refiere a la propiedad de que el error de representación disminuye al aumentar el grado del polinomio (en el caso de las bases reducidas aumenta el número de elementos en la base). Por otro lado el término de *refinamiento h* se toma prestado de los métodos de diferencias finitas, donde el tamaño de cada celda de la grilla es representado por h. En este caso el refinamiento ocurre en el espacio de los parámetros (y no en el dominio físico).

2.2.1. Refinamiento h

Partiendo de la siguiente notación:

- V: espacio de parámetros para un dado subdominio D.
- V_1, V_2 : particiones de V.
- Λ_V : parámetros *greedy* para V.
- $\hat{\Lambda}_V$: punto de anclaje para V.

El refinamiento en el dominio de los parámetros ocurre a partir de la división recursiva de cada subdominio V del dominio total D en dos subdominios V_1 y V_2. De forma que se obtiene una estructura de árbol binario.

Esta descomposición binaria del dominio está descrita en forma de pseudocódigo en el algoritmo 2.

Al algoritmo ingresan tres objetos:

- λ_V: conjunto de parámetros resultado de un muestreo de V.
- $\hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2}$: puntos de anclaje (son los primeros dos elementos de Λ_V).

Luego, para cada parámetro del conjunto λ_V se evalúa su distancia a los puntos de anclaje a partir de la *función de proximidad* $d : d(\lambda_1, \lambda_2)$:

$$d(\lambda_1, \lambda_2) = ||\lambda_1 - \lambda_2||_2,$$

de forma que se obtengan dos conjuntos; λ_{V_1} con los λ_i más proximos a $\hat{\Lambda}_{V_1}$, y λ_{V_2} con los λ_i más proximos a $\hat{\Lambda}_{V_2}$, tal que $\lambda_V = \lambda_{V_1} \cup \lambda_{V_2}$. Este resultado es la división del espacio de parámetros a partir de los puntos de anclaje.

Algoritmo 2 `Partition`$(\lambda_V, \hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2})$

Input: $\lambda_V, \hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2}$

$\lambda_{V_1} = \lambda_{V_2} = \emptyset$
for each $\lambda_i \in \lambda_V$ **do**
  if $d(\lambda_i, \hat{\Lambda}_{V_1}) < d(\lambda_i, \hat{\Lambda}_{V_2})$ **then**
    $\lambda_{V_1} = \lambda_{V_1} \cup \lambda_i$
  else if $d(\lambda_i, \hat{\Lambda}_{V_1}) > d(\lambda_i, \hat{\Lambda}_{V_2})$ **then**
    $\lambda_{V_2} = \lambda_{V_2} \cup \lambda_i$
  else
    $\lambda_{V'} = \text{random choice}([\lambda_{V_1}, \lambda_{V_2}])$
    $\lambda_{V'} = \lambda_{V'} \cup \lambda_i$
  end if
end for

Output: $\lambda_{V_1}, \lambda_{V_2}$

2.2.2. Refinamiento hp-greedy

El refinamiento hp-greedy es un método que combina el algoritmo greedy para la construcción de bases reducidas con la partición del dominio de parámetros.

Esta partición recursiva del dominio de parámetros da lugar a una estructura de árbol binario, la cual tendrá diferentes niveles l de profundidad, con un l_{max} establecido por el usuario, de forma que $l : 0 \leq l \leq l_{max}$, donde $l = 0$ es el nodo raíz. Cada nodo del árbol estará etiquetado por un conjunto de índices B_l, que parte de:

$$B_0 = (0,),$$

luego sus dos hijos ($l = 1$) tendrán las etiquetas:

$$B_1 = (0, 0,) \quad o \quad (0, 1,),$$

y en general:

$$B_l = (0, i_1, \cdots, i_l), \quad con \quad i_j = \{0, 1\},$$

donde cada nivel l tendrá un máximo de 2^l nodos. Los nodos que no tengan hijos se llamarán nodos *hojas*.

El método está explicado en el algoritmo 3; partiendo de un dado dominio de parámetros V se construye una base reducida a partir de un conjunto de entrenamiento $\mathcal{T}_V = \{\lambda_{V_i}, h_{\lambda_{V_i}}\}_{i=1}^{N}$, una *tolerancia greedy* ε y un n_{max} (para esto se utiliza el algoritmo 1). Si el error de representación σ es mayor que la tolerancia ε, y si la profundidad del nivel l es menor a l_{max}, entonces se realizará una partición del dominio V utilizando como puntos de anclaje a los dos primeros parámetros *greedy*. En cada dominio se realizará el mismo procedimiento hasta que se cumpla que $l = l_{max}$ o hasta que $\sigma \leq \varepsilon$.

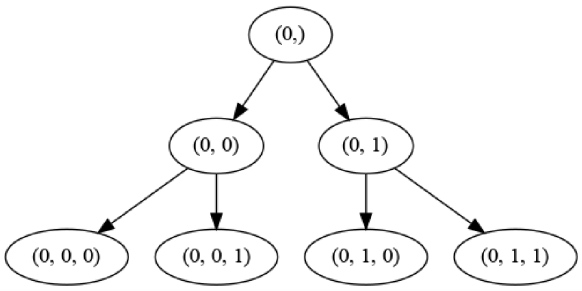

Figura 2.2: Representación de los nodos de un árbol con $l_{max} = 2$ [14].

Algoritmo 3 `hpGreedy`$(\mathcal{T}, \varepsilon, n_{max}, l, l_{max}, B_l)$

Input: $\mathcal{T} = \{\lambda_i, h_i\}_{i=0}^{N}, \varepsilon, n_{max}, l, l_{max}, B_l$

rb, Λ_V, σ = `GreedyRB`$(\mathcal{T}_V, \lambda_V, \varepsilon, n_{max})$
if $\sigma > \varepsilon$ **and** $l < l_{max}$ **then**
    $\hat{\Lambda}_{V_1} = \Lambda_V[1]$
    $\hat{\Lambda}_{V_2} = \Lambda_V[2]$
    $\lambda_{V_1}, \lambda_{V_2}$ = `Partition`$(\lambda_V, \hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2})$
    out_1 = `hpGreedy`$(\mathcal{T}_{V_1}, \lambda_{V_1}, \varepsilon, n_{max}, l+1, l_{max}, (B_l, 0))$
    out_2 = `hpGreedy`$(\mathcal{T}_{V_2}, \lambda_{V_2}, \varepsilon, n_{max}, l+1, l_{max}, (B_l, 1))$
    $out = out_1 \cup out_2$
else
    $out = \{(rb, \Lambda_V, B_l)\}$
end if

Output: out

El resultado del algoritmo 3 es una estructura arbórea, donde cada nodo contiene la información de sus puntos de anclaje, por lo que en el caso de querer proyectar un conjunto de validación, cada onda gravitacional se proyectará a la base reducida del nodo hoja con el punto de anclaje más cercano al parámetro de la onda.

2.2.3. Aplicación a Ondas Gravitacionales

Se trabaja a partir de un conjunto de ondas gravitacionales con parámetro bidimensional, donde $\chi_{1_z} = \chi_{2_z} = \chi_z$, es decir que $\lambda = (q, \chi_z)$. De esta forma se puede graficar fácilmente el dominio de parámetros.

En la figura 2.3 se puede observar una representación de la partición del dominio de parámetros. En la primera imagen se pueden ver los dos puntos de anclaje, que son los primeros dos elementos de la base global construida inicialmente. En cada nueva división se construye una nueva base global con la cual se realiza la siguiente partición de cada subdominio.

En la figure 2.4 se compara el máximo error de representación obtenido para un conjunto de validación con una base global, es decir, con $l_{max} = 0$, y con $l_{max} = 4$. La

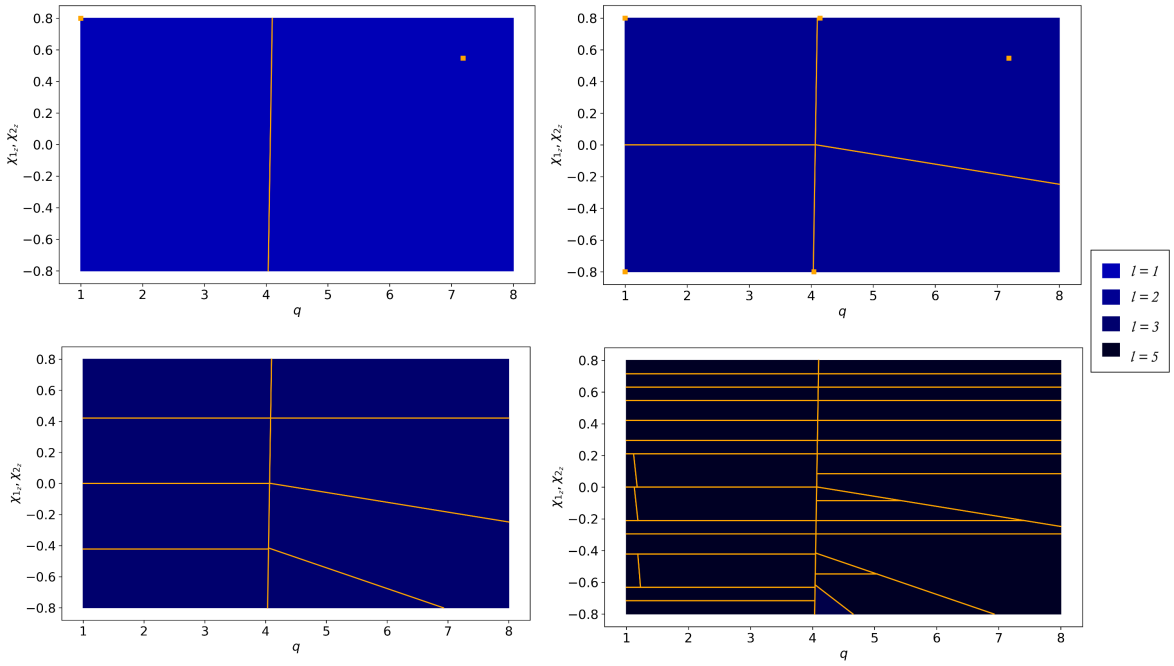

Figura 2.3: Ejemplo de partición del espacio de parámetros bidimensional para $l_{max} = 1, 2, 3$ y 5. En los primeros dos casos se muestran los puntos de anclaje.

velocidad de convergencia es claramente mayor en el segundo caso.

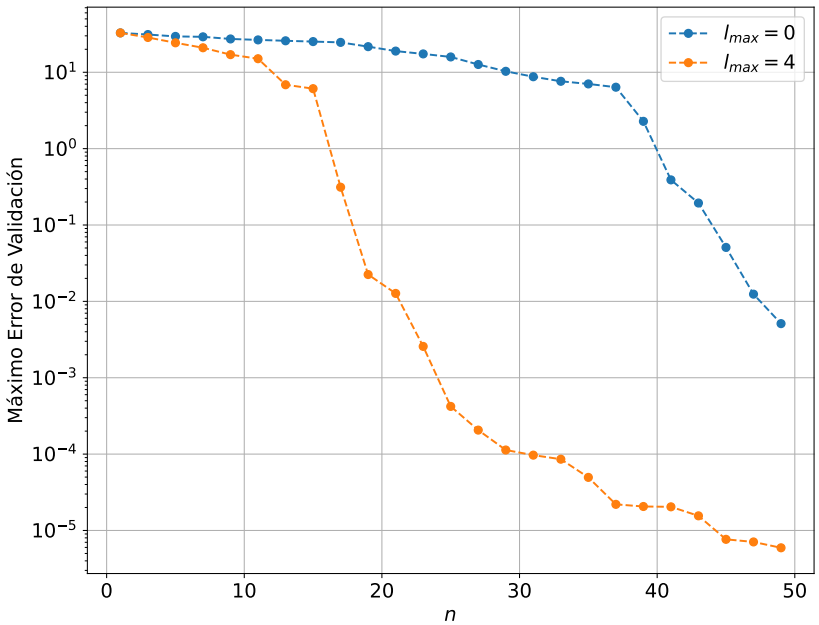

Figura 2.4: Base global ($l_{max} = 0$) versus base con $l_{max} = 4$ para distintos valores de n.

El aspecto más importante de este método es que permite disminuir la complejidad temporal del algoritmo a la hora de proyectar la base, a cambio de aumentar la complejidad espacial, pues si bien cada subdominio tendrá un máximo de n_{max} elementos

en su base, habrá un máximo de $2^{l_{max}}$ subdominios.

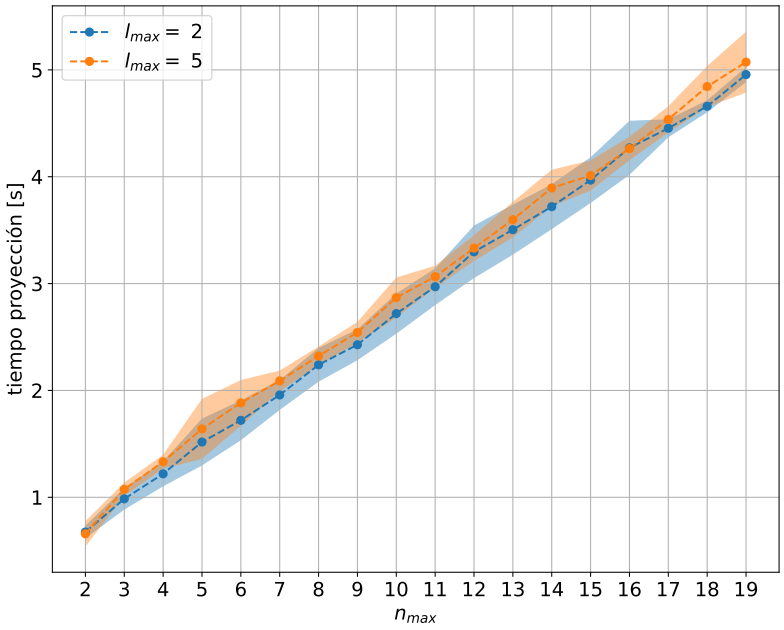

Figura 2.5: tiempos de proyección de un conjunto de validación a dos bases con distinto l_{max} en función del n_{max}. En cada caso la linea de trazo representa el valor medio, y el área de color indica una desviación estandar desde el valor medio, para cada medición.

En la figura 2.5 se graficó el tiempo de proyección de un conjunto de validación a dos bases *hp-greedy* con distinto valor de l_{max}. Se observa que el tiempo es bastante lineal en relación al n (elementos de las bases locales), y no parece ser afectado por l_{max}.

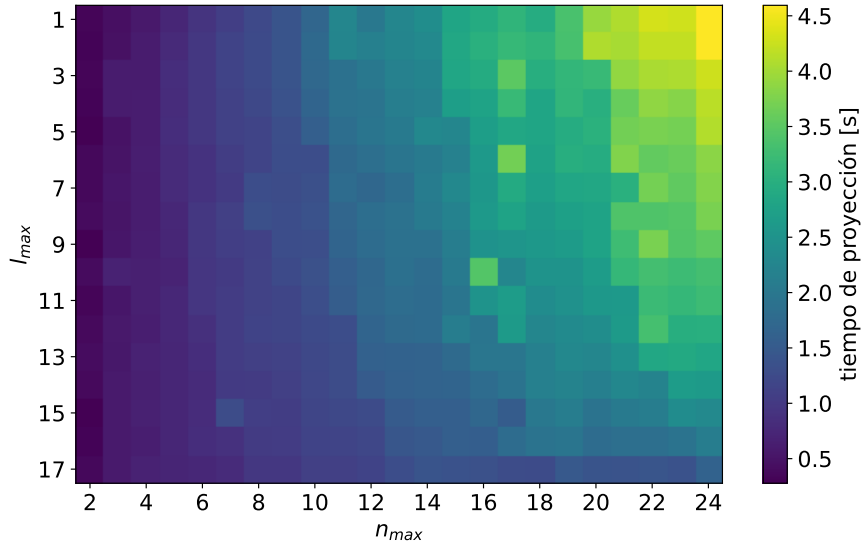

Figura 2.6: Tiempo de proyección de un conjunto de validación para diferentes valores de n_{max} y l_{max}

En la figura 2.6 se puede ver el tiempo de proyección para más valores de n_{max} y l_max. En los primeros valores de l_{max} se observa un comportamiento similar al descrito

anteriormente, donde el tiempo depende casi únicamente de n_{max}. Sin embargo al aumentar el l_{max} se observa que el tiempo disminuye. Esto se puede entender en dos partes:

- **Independencia aparente entre el tiempo de proyección y** l_{max}: para realizar la proyección de cada onda del conjunto de validación en la base *hp-greedy* primero se debe buscar el subdominio (la hoja) correspondiente utilizando los puntos de anclaje de la estructura arbórea de la base. Luego se proyectará la onda en la base local del subdominio en cuestión. Si bien la búsqueda en el árbol tiene una complejidad temporal $O(l_{max})$, el trabajo de cómputo más importante es el que se realizará al momento de proyectar la base, que es independiente de l_{max}, con una complejidad temporal $O(n_{max})$. Pero esto solo se cumple hasta ciertos valores de l_{max}.

- **Disminución del tiempo de proyección al aumentar** l_{max}: ya se mencionó en más de una ocasión que por cada nivel l hay un máximo de 2^l subdomínios. Es decir que si se quiere obtener el número de elementos de todas las bases en las hojas del árbol, suponiendo un árbol denso, este número será $n_{max} \times 2^{l_{max}}$. En la figura 2.6 se utilizó un conjunto de entrenamiento con 1400 ondas, por lo que al llegar a unos valores de $l_{max} = 6$ y $n_{max} = 24$ en total debería haber 1536 elementos de base en total. Es decir, más elementos de base que ondas en el conjunto de entrenamiento. Por lo tanto al aumentar el l_{max} rápidamente se aumenta el número de subdominios, reduciendo su tamaño como resultado y reduciendo el número de elementos de las bases locales (cada subdominio tendrá una cantidad de elementos menor a n_{max}). De esta forma se explica la disminución del tiempo de proyección para valores grandes de l_{max}, consecuencia del tamaño limitado del conjunto de entrenamiento.

2.2.4. Hiperparámetros

Al momento de construir una base *hp-greedy* entran en juego cuatro hiperparámetros. Los primeros tres son los parámetros de parada;

- $\boldsymbol{n_{max}}$: determina la cantidad máxima de elementos para cada base local. A mayor cantidad de elementos el error de representación será menor, pero el tiempo requerido para proyectar un conjunto de validación a la base depende casi exclusivamente de este hiperparámetro.

- $\boldsymbol{l_{max}}$: determina la máxima profundidad de las hojas del árbol. En general al aumentar l_{max} disminuye el error de representación, pero valores muy elevados

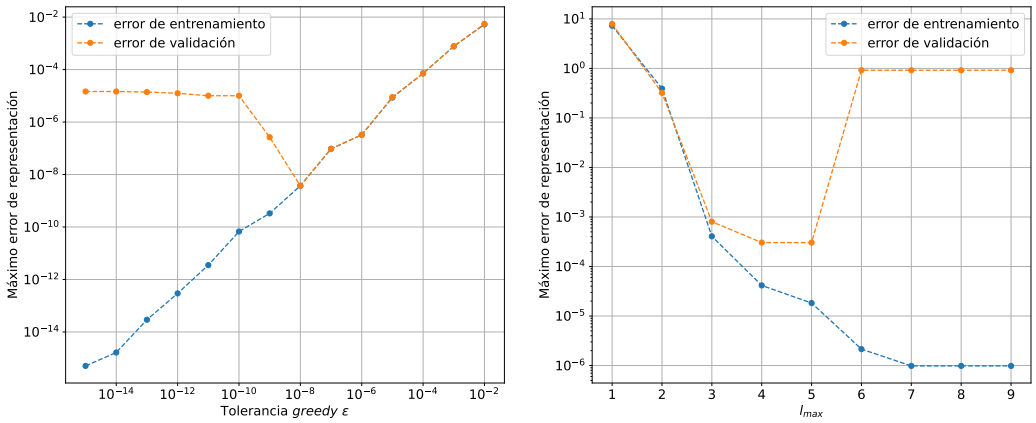

Figura 2.7: Ejemplos de *sobreajuste*. A la izquierda variando ε con $(n_{max}, l_{max}) = (25, 19)$ en un conjunto de parámetro unidimensional $(\lambda_i = q_i)$. A la derecha variando l_{max} con $(n_{max}, \varepsilon) = (20, 1 \times 10^{-6})$ en un conjunto de parámetro bidimensional$(\lambda_{ij} = (q_i, \chi_{z_j}))$.

junto a cierta combinación de hiperparámetros pueden dar lugar a sobreajustes en el modelo, un ejemplo de esto se puede ver en la figura 2.7. Este es un comportamiento típico de las estructuras arbóreas.

- $\boldsymbol{\varepsilon}$: la tolerancia *greedy* interviene tanto en el tamaño de las bases locales como en la profundidad de las hojas del árbol. Un valor de ε demasiado bajo también puede dar lugar a sobreajuste, sobre todo con valores muy altos de l_{max}. Un valor de $\varepsilon = 0$ implica que se obtiene un árbol totalmente denso, determinado únicamente por n_{max} y l_{max}, y al aumentar el valor de ε se puede pensar en la analogía de podar un árbol, de forma que se previene el sobreajuste.

Al cuarto hiperparámetro se le da el nombre de ***semilla*** y en este trabajo se la denota con $\hat{\mathbf{\Lambda}}_\mathbf{0}$ para diferenciarla de la semilla de una base local, denotada por Λ_1;

- $\hat{\mathbf{\Lambda}}_\mathbf{0}$: la semnilla no es más que el primer parámetro *greedy* de la base global. En cada base local, el primer parámetro *greedy* no es relevante, pero en el caso de las bases *hp-greedy* cada semilla dará lugar a una división diferente del dominio de parámetros. En la figura 2.8 se puede ver como cuatro semillas diferentes dan lugar a cuatro curvas de error con distinta convergencia. En la figura 2.9, por otro lado, se observa el resultado de la partición del dominio para tres semillas diferentes. En general las semillas que mejor funcionan (con el conjunto de datos utilizado) son las que logran una partición regular del dominio de parámetros.

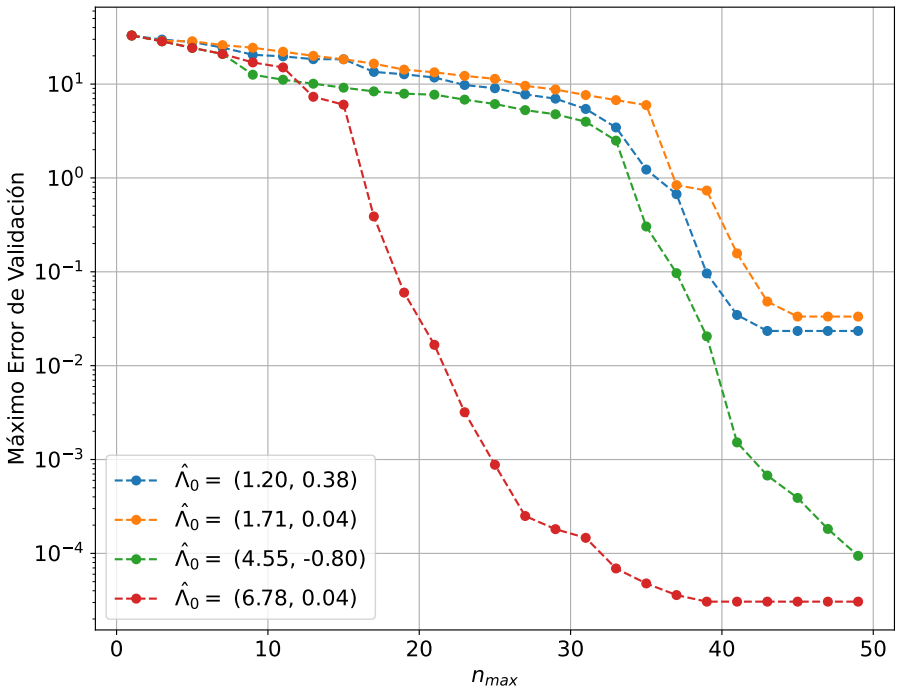

Figura 2.8: Error de validación para diferentes semillas.

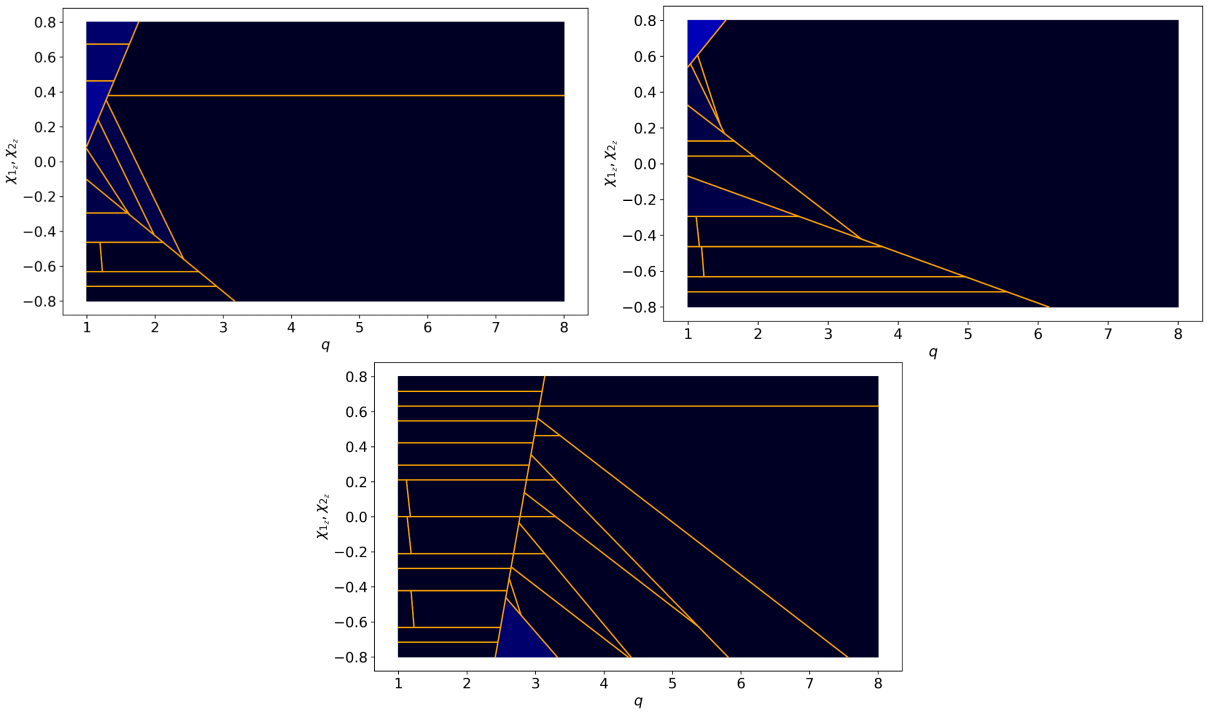

Figura 2.9: Partición del espacio de parámetros para tres semillas diferentes; a la izquierda $\hat{\Lambda}_0 = (1{,}2 \;\; 0{,}38)$, a la derecha $\hat{\Lambda}_0 = (1{,}71 \;\; 0{,}04)$ y al centro $\hat{\Lambda}_0 = (4{,}55 \;\; -0{,}8)$.

Capítulo 3

Optimización de Hiperparámetros

3.1. Planteo del Problema

Sea $f : X \to \mathbb{R}$ una función que devuelve el máximo error de validación de un modelo entrenado a partir de una combinación de hiperparámetros $\mathbf{x} \in X$, se desea encontrar $\hat{\mathbf{x}}$:

$$\mathbf{x}^* = arg\min_{\mathbf{x} \in X} f(\mathbf{x})$$

Es decir, se busca encontrar la combinación óptima de hiperparámetros dentro de un dominio X para obtener el mínimo error de representación en un dado conjunto de validación. En el caso de la construcción de una base *hp-greedy* óptima:

$$\mathbf{x} = (n_{max}, l_{max}, \varepsilon, \hat{\Lambda}_0).$$

El problema al momento de realizar esta optimización es que la función f no tiene una expresión analítica, sino es que es el resultado de entrenar el modelo y evaluar el error de representación con un conjunto de validación, lo que la hace costosa de evaluar (computacionalmente hablando). Este capítulo se centrará principalmente en la **optimización Bayesiana** [23, 24], un método que intenta reducir al mínimo el número de evaluaciones de f para encontrar $\hat{\mathbf{x}}$ y se puede colocar dentro de una categoría llamada optimización secuencial basada en modelos, o **SMBO**[25, 26] (*Secuential Model-Based Optimization*).

Además existen dos métodos muy utilizados que no utilizan modelos, los cuales son la **busqueda exaustiva** (o *grid search*) y la **búsqueda aleatoria**. Estos métodos se utilizaron en casos sencillos de optimización para realizar una comparación con la optimización bayesiana.

Comentario sobre el dominio X

Si bien la tolerancia *greedy* ε puede tomar cualquier valor real no nulo (a diferencia de n_{max}, l_{max} y $\hat{\Lambda}_0$ que toman valores discretos), para simplificar la búsqueda de $\hat{\mathbf{x}}$ se utilizaron siempre distribuciones discretas en el espacio logarítmico. Más específicamente se utilizaron conjuntos de la forma $C = \{1 \times 10^t \mid a \leq t \leq b, t \in \mathbb{Z}\}$. De esta forma X será un conjunto finito y estará definido por los valores extremos de cada hiperparámetro.

3.2. Optimización Bayesiana

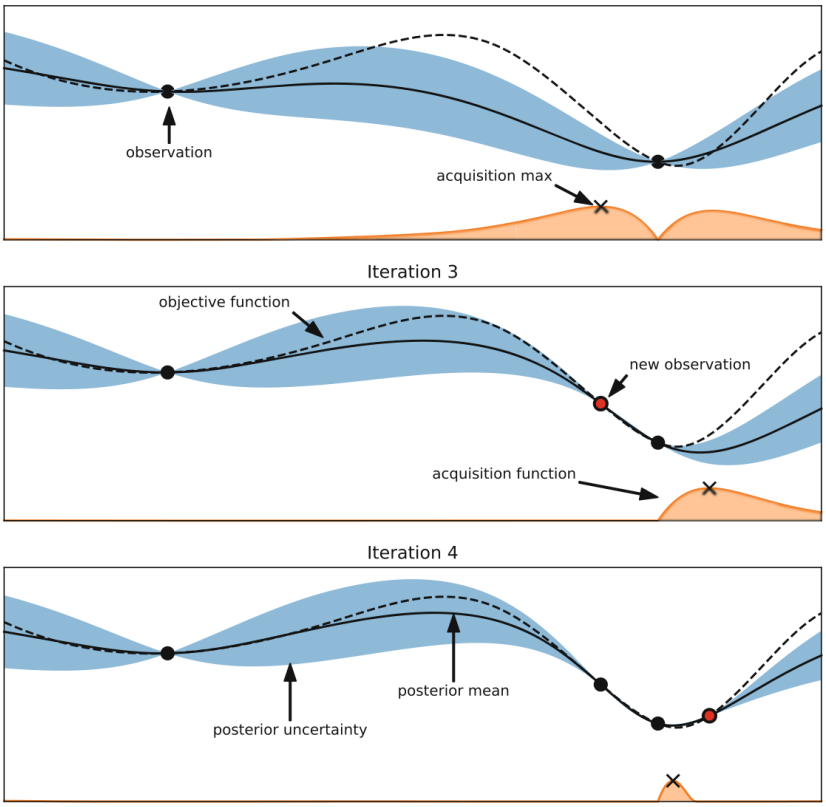

Figura 3.1: En la figura se observan tres iteraciones de una optimización bayesiana para una función sencilla con parámetro unidimensional. En linea punteada está representada la función real, mientras que con linea gruesa se representa el valor medio del modelo estadístico (en este caso construido utilizando procesos gaussianos). El área pintada en azul representa la incertidumbre del modelo, que tiende a cero en los puntos que representan las observaciones realizadas. Debajo se puede ver una función de adquisición en color naranja, que indica el siguiente punto a evaluar [27].

La optimización bayesiana es un método que utiliza la información de todas las evaluaciones realizadas de la función f para decidir que valor de $\mathbf{x}$ evaluar a continuación, reduciendo así el número necesario de evaluaciones de f para encontrar el mínimo.

Para explicar como funciona este método se parte de un formalismo llamado optimización secuencial basada en modelos, que no es más que una generalización de la

optimización bayesiana.

3.2.1. Optimización Secuencial Basada en Modelos

La idea es aproximar la función f a partir de un modelo sustituto $\mathcal{M}$.

Se parte de un conjunto de observaciones $D = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(k)}, y^{(k)})\}$, donde $y^{(j)} = f(\mathbf{x}^{(j)})$, a partir del cual se ajusta el modelo sustituto $\mathcal{M}$. Luego utilizando las predicciones del modelo se maximiza una función S llamada función de adquisición que elije el siguiente conjunto de hiperparámetros $\mathbf{x}_i \in X$ para evaluar la función f y se agrega el par $(\mathbf{x}_i, f(\mathbf{x}_i))$ al conjunto de observaciones D. Una vez hecho esto se vuelve a ajustar el modelo $\mathcal{M}$ y se repite el proceso, que está explicado en forma de pseudocódigo en el algoritmo 4.

Algoritmo 4 `SMBO`

Input: $f, X, S, \mathcal{M}$

1: $D = \text{InicializarMuestras}(f, X)$
2: **for** $i = 1, 2, ...$ **do**
3: $\quad \mathcal{M} = \text{AjustarModelo}(D)$
4: $\quad \mathbf{x}_i = arg\,\text{máx}_{\mathbf{x} \in X}\, \mathcal{S}(\mathbf{x}, \mathcal{M})$.
5: $\quad y_i = f(\mathbf{x}_i)$ ▷ Paso costoso
6: $\quad D = D \cup \{(\mathbf{x}_i, y_i)\}$
7: **end for**

Optimización Bayesiana

Lo que caracteriza a la optimización bayesiana dentro del formalismo de la optimización secuencial basada en modelos, es justamente la creación del modelo. En la optimización bayesiana se construye un modelo estadístico, donde se representa con $P(y|\mathbf{x})$ la predicción del modelo, siendo y el resultado de una evaluación $f(\mathbf{x})$. El nombre del método se debe a que para la construcción del modelo se utiliza el teorema de Bayes:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)\ P(y)}{P(\mathbf{x})}$$

En la terminología bayesiana, se conoce a $P(y|\mathbf{x})$ como probabilidad a posteriorí o *posterior*, que es proporcional a la probabilidad a priori o *prior* $P(y)$ por la función de verosimilitud o *likelihood* $P(\mathbf{x}|y)$. La probabilidad $P(\mathbf{x})$ es una probabilidad marginal que sirve como factor de normalización, por lo que no es de tanto interés.

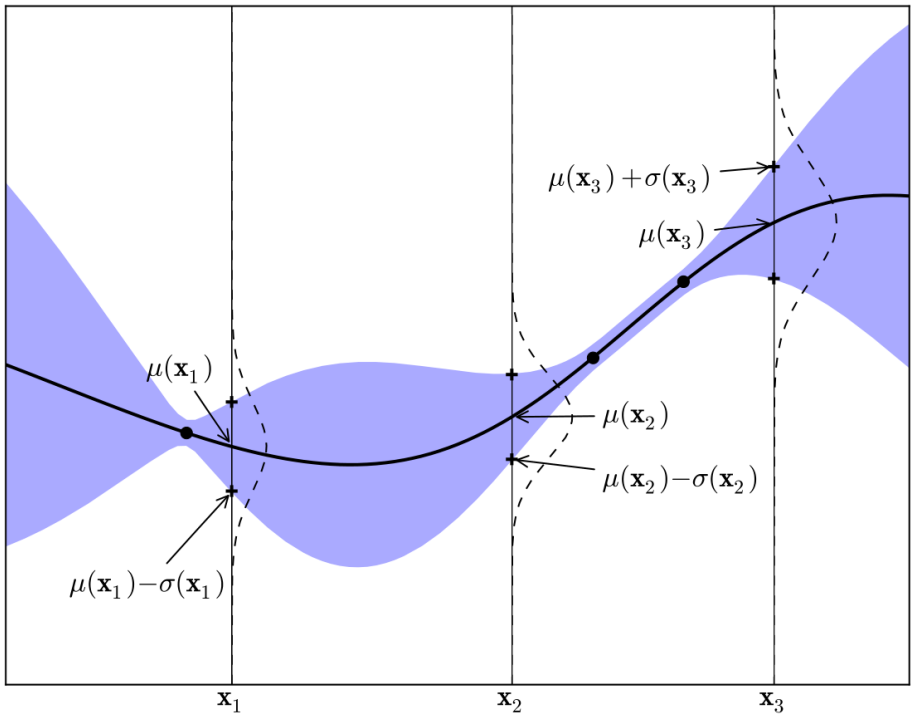

Figura 3.2: Proceso Gaussiano unidimensional con tres observaciones representadas por los puntos negros. La linea gruesa representa la media del modelo predictivo y la zona azul la varianza en cada caso. Se representa con linea de trazo las distribuciones normales para los valores x_1, x_2, y x_3[24].

Procesos Gaussianos

Una opción muy utilizada para la construcción del *prior* y actualización del *posterior* son los procesos gaussianos. Una forma sencilla de entender un proceso gaussiano es pensarlo como una función que para cada valor de x devuelve la media $\mu(x)$ y la varianza $\sigma(x)$ de una distribución normal, en el caso particular de que x sea unidimensional (ver figura 3.2). Con $\mathbf{x}$ multidimensional, se obtiene una distribución normal multivariante, caracterizada por el vector $\boldsymbol{\mu}(\mathbf{x})$ y la matriz de covarianza $\Sigma(\mathbf{x}, \mathbf{x}')$.

Sin embargo en este trabajo no se utilizan procesos gaussianos, principalmente porque parten del supuesto de que f es continua. Para una introducción a la optimización bayesiana con procesos gaussianos ver [24].

3.2.2. Mejora Esperada: Función De Adquisición

Para la elección de los puntos a evaluar en la función real se maximiza la función de adquisición S. Existen varias propuestas de funciones de adquisición, pero en este caso se utiliza la **mejora esperada** o EI(*Expected Improvement*) [28]. Sea y^* un valor de referencia, se define a la mejora esperada con respecto a y^* como:

$$EI_{y^*}(\mathbf{x}) := \int_{-\infty}^{\infty} \text{máx}(y^* - y, 0) p(y|\mathbf{x})\ dy \tag{3.1}$$

3.2.3. Estimador de Parzen con Estructura Arbórea

El estimador de Parzen con estructura arbórea o **TPE** (*Tree-Structured Parzen Estimator*) [26] es una estrategia que modela $P(x_i|y)$ para cada $x_i \in X_i$ (es decir, que x_i representa a cada hiperparámetro por separado) a partir de dos distribuciones creadas a utilizando las observaciones D:

$$P(x_i|y) = \begin{cases} \ell(x_i) & \text{si } y < y^* \\ g(x_i) & \text{si } y \geq y^*, \end{cases} \tag{3.2}$$

Donde las densidades $\ell(x_i)$ y $g(x_i)$ se construyen a partir de dos conjuntos D_ℓ y D_g, ambos subconjuntos de D, tal que D_ℓ contiene todas las observaciones con $y < y^*$, y D_g contiene a todo el resto de forma que $D = D_\ell + D_g$. El valor de referencia y^* será un valor por encima del mejor valor observado de $f(\mathbf{x})$, que se selecciona para ser un cuantil $\gamma \in (0, 1)$ de los valores observados y tal que $P(y < y^*) = \gamma$.

Mejora Esperada con TPE

Aplicando la ecuación (3.2) a la definición de mejora esperada (3.1) se obtiene la siguiente relación [26]:

$$EI_{y^*}(x_i) \propto \left(\gamma + (1-\gamma)\frac{g(x_i)}{\ell(x_i)}\right)^{-1} \tag{3.3}$$

Es decir que para maximizar la mejora esperada se debe escoger un valor x_i que maximice el cociente $\ell(x_i)/g(x_i)$ (o minimice $g(x_i)/\ell(x_i)$).

Estimación de las Densidades

Las densidades de probabilidad se estiman utilizando ventanas de Parzen. Sea $D_x = \{x_i \mid (\mathbf{x}, y) \in D_\ell \ (o \ D_g)\}$:

$$P(x_i) = \frac{\sum_{x'_i \in D_x} w_{x'_i} k(x_i, x'_i) + w_p k(x_i, x_p)}{\sum_{x'_i \in D_x} w_{x'_i} + w_p}, \tag{3.4}$$

donde $w_{x'_i}$ es el peso de la observación x'_i(ver [29]), x_p es un valor fijo *prior*, w_p es un peso *prior* (por defecto igual a 1) y k es la función *kernel*, que en este caso son distribuciones gaussianas truncadas centradas en los puntos x'_i (ver [30] para más detalle).

Algoritmo

Finalmente se puede ver el procedimiento completo del estimador de Parzen con estructura arbórea en el algoritmo 5. Un detalle importante es que el algoritmo requie-

re un valor n_c, que es el número de candidatos que se utilizarán para maximizar el cociente $\ell(x_i)/g(x_i)$ (es decir, para maximizar la función de adquisición). En la linea 6 del algoritmo se realiza el muestro de los n_c candidatos, utilizando la distribución $\ell(x_i)$, para luego seleccionar x_i^* a partir del conjunto C_i. Para este trabajo se utilizó la implementación de este algoritmo realizada en el paquete ***Optuna*** [31] escrito en el lenguaje de programación Python.

Algoritmo 5 `TPE`

Input:
$D = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(k)}, y^{(k)})\}$ ▷ Observaciones
$n_t \in \mathbb{N}$ ▷ número de iteraciones
$n_c \in \mathbb{N}$ ▷ número de candidatos
$\gamma \in (0, 1)$ ▷ cuantil para obtener y^*

1: **for** $t = 1, 2, ..., n_t$ **do**
2: $D_\ell = \{(\mathbf{x}, y) \in D | y < y^*, \text{ con } P(y < y^*) = \gamma\}$
3: $D_g = D - D_\ell$
4: **for** $x_i = n_{max}, l_{max}, \varepsilon, ...$ **do** ▷ Para cada hiperparámetro
5: Construir $\ell(x_i)$ con $\{x_i \mid (\mathbf{x}, y) \in D_\ell\}$ y $g(x_i)$ con $\{x_i \mid (\mathbf{x}, y) \in D_g\}$.
6: $C_i = \{x_i^{(j)} \sim \ell(x_i) | j = 1, ..., n_c\}$ ▷ muestreo de n_c candidatos para x_i^*
7: $x_i^* = arg\,\text{máx}_{x_i \in C_i}\, \ell(x_i)/g(x_i)$
8: **end for**
9: $D = D \cup \{(\mathbf{x}^*, f(\mathbf{x}^*)\}$ ▷ $\mathbf{x}^*$ es el vector construido a partir de cada x_i.
10: **end for**

Output: $\mathbf{x}$ con el mínimo valor y en D.

TPE Multivariante

Una alternativa al algoritmo 5 es el algoritmo **TPE Multivariante**, implementado en Optuna [1]. La única diferencia es que en este caso las densidades no se construyen para cada hiperparámetro por separado, sino que se utilizan ventanas de Parzen multivariadas, dónde las funciones *kernel* ahora son distribuciones gaussianas multivariadas. Es decir que en lugar de construir $\ell(x_i)$ y $g(x_i)$ para cada hiperparámetro, ahora se construye directamente $\ell(\mathbf{x})$ y $g(\mathbf{x})$.

3.3. Optimización Multiobjetivo

En este sección se explicará de forma básica el funcionamiento del algoritmo MOTPE (*Multiobjective Tree-Structured Parzen Estimator*) [30, 32], sin entrar en demasiados detalles, pues si bien es un método interesante, se puede conseguir un mejor resultado utilizando el TPE clásico, y limitando el máximo valor de n_{max} permitido.

[1] El algoritmo TPE Multivariante fue introducido en la siguiente actualización de Optuna: https://github.com/optuna/optuna/pull/1767

3.3.1. Planteo del Problema

Minimizar el error de representación no es el único objetivo posible al momento de construir una base reducida *hp greedy*. También es de mucho interés minimizar el tiempo necesario para proyectar un conjunto de validación más denso a la base ya creada. Esto se puede plantear como el problema de minimizar la función $\mathbf{f}(\mathbf{x})$:

$$\min_{\mathbf{x}\in X} \mathbf{f}(\mathbf{x}) := (f_1(\mathbf{x}), f_2(\mathbf{x}))$$

Cuando se quiere minimizar dos o más objetivos a la vez aparece el problema de que estos entran en conflicto entre sí, por lo que ya no se puede hablar de encontrar un valor mínimo en general, pero sí se puede encontrar un conjunto de elementos llamado **frente de Pareto** formado por los mejores valores encontrados. Con el conjunto de Pareto se decidirá que obetjivo es más importante relativo al resto, y se elegirá el valor deseado.

Para entender mejor esto y el algoritmo MOTPE se empezará con algunas definiciones matemáticas que serán relevantes.

3.3.2. Preliminares Matemáticos

Definición 1 Relación de Dominancia. Un vector $\mathbf{y} \in \mathbb{R}^n$ domina al vector $\mathbf{y'} \in \mathbb{R}^n$ si y solo si $\forall i : y_i \leq y'_i$ y $\exists i : y_i < y'_i$, y se denota con $\mathbf{y} \prec \mathbf{y'}$. Un vector $\mathbf{y} \in \mathbb{R}^n$ domina débilmente al vector $\mathbf{y'} \in \mathbb{R}^n$ si y solo si $\forall i : y_i \leq y'_i$, y se denota con $\mathbf{y} \preceq \mathbf{y'}$.

Definición 2 Relación incomparable. Dos vectores $\mathbf{y}, \mathbf{y'} \in \mathbb{R}^n$ son incomparables si y solo si no se cumple que $\mathbf{y} \preceq \mathbf{y'}$ ni que $\mathbf{y'} \preceq \mathbf{y}$, y se denotan $\mathbf{y}||\mathbf{y'}$.

Definición 3 Relación de dominancia entre un vector y un conjunto. Para un conjunto finito de vectores $Y \subset \mathbb{R}^n$ y un vector $\mathbf{y} \in \mathbb{R}^n$, se define $Y \prec \mathbf{y}$ ($Y \preceq \mathbf{y}$) si y solo si $\exists\ \mathbf{y'} \in Y_{rango(1)} : \mathbf{y'} \prec \mathbf{y}$ ($\mathbf{y'} \preceq \mathbf{y}$). También se define $\mathbf{y} \prec Y$ ($\mathbf{y} \preceq Y$) si y solo si $\exists\ \mathbf{y'} \in Y_{rango(1)} : \mathbf{y} \prec \mathbf{y'}$ ($\mathbf{y} \preceq \mathbf{y'}$).

Definición 4 Relación Incomparable entre un vector y un conjunto. Para un conjunto finito de vectores $Y \subset \mathbb{R}^n$ y un vector $\mathbf{y} \in \mathbb{R}^n$, se define que $Y||\mathbf{y}$ (equivalente a $\mathbf{y}||Y$) si y solo si $\forall \mathbf{y'} \in Y_{rango(1)} : \mathbf{y}||\mathbf{y'}$.

Definición 5 Óptimo de Pareto. Dada una función $\mathbf{f} : X \to \mathbb{R}^n$, un vector $\mathbf{x} \in X$ es óptimo de Pareto si y solo si $\nexists \mathbf{x'} \in X : \mathbf{f(x')} \prec \mathbf{f(x)}$. Un conjunto de vectores óptimos de Pareto $\{\mathbf{x} \in X | \nexists \mathbf{x'} \in X : \mathbf{f(x')} \prec \mathbf{f(x)}\}$ se llama conjunto de Pareto. El conjunto de las imagenes del conjunto de Pareto $\{\mathbf{f(x)} \in \mathbb{R}^n | \mathbf{x} \in X \text{es óptimo de Pareto}\}$ se llama

frente de Pareto.

Lo más importante de esta subsección es entender la relación de dominancia y el concepto del frente de Pareto.

3.3.3. Estimador de Parzen Multiobjetivo con Estructura Arbórea

El MOTPE es una extensión del TPE clásico, adaptado para optimizar una función multiobjetivo.

Dado el conjunto de observaciones $D = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \cdots, (\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\}$, donde $\mathbf{y}^{(j)} = \mathbf{f}(\mathbf{x}^{(j)})$, se modela $P(x_i|\mathbf{y})$ para cada hiperparámetro x_i usando las funciones de densidad de probabilidad $\ell(x_i)$ y $g(x_i)$:

$$P(x_i|\mathbf{y}) = \begin{cases} \ell(x_i) & \text{si } (\mathbf{y} \prec Y^*) \vee (\mathbf{y} \| Y^*) \\ g(x_i) & \text{si } Y* \preceq \mathbf{y}, \end{cases} \tag{3.5}$$

con Y^* un conjunto construido tal que $p((\mathbf{y} \prec Y^*) \vee (\mathbf{y} \| Y^*)) = \gamma$. Nuevamente, $\gamma \in (0, 1)$ es un cuantil, y las funciones $\ell(x_i)$ y $g(x_i)$ se construyen utilizando ventanas de Parzen en base a las observaciones realizadas.

En este caso la función de **mejora esperada** es un poco más complicada, pero se reduce al mismo resultado que con el TPE clásico; para maximizar la mejora esperada se debe maximizar la relación $\ell(x_i)/g(x_i)$ para cada x_i. Para ver en mayor detalle este resultado y el algoritmo completo del método, se recomienda ver [32].

Capítulo 4

Resultados

En este capítulo se recogen los resultados más relevantes de la optimización de Hiperparámetros

4.1. Optimización del Máximo Error de Validación

En esta sección se encuentran los resultados de las optimizaciones realizadas teniendo en cuenta únicamente el máximo error de validación

4.1.1. Conjunto pequeño: Comparación de métodos

Utilizando un conjunto de entrenamiento con cien ondas equidistantes en el espacio del parámetro unidimensional $q : 1 < q < 8$, se quiere optimizar el error de representación para un conjunto de validación con quinientas ondas (cinco veces más denso). Los hiperparámetros a optimizar son $\mathbf{x} = (n_{max}, l_{max}, \hat{\Lambda}_0)$, dejando ε fijo en 1×10^{-12} para simplificar la búsqueda, que se realiza en los siguientes intervalos:

$$
\begin{aligned}
n_{max} &\in \{5, 6, 7, ..., 20\}, \\
l_{max} &\in \{1, 2, 3, ..., 10\}, \\
\hat{\Lambda}_0 &\in \{q_0 \mid q_0 = 1 + i\Delta q,\ i \in \mathbb{N} : 0 \leq i \leq 99,\ \Delta q = 7/99\}.
\end{aligned}
$$

Son 16 valores de n_{max}, 10 valores de l_{max} y 100 para q_0 ($\hat{\Lambda}_0 = q_0$). Lo que hace un total de 16000 combinaciones posibles.

En la figura 4.1 se puede ver el resultado de realizar 20 optimizaciones de 100 iteraciones con tres diferentes métodos; búsqueda aleatoria (*random*), TPE y TPE multivariante (los tres métodos están implementados en Optuna [31]). En linea oscura se representa la media del mejor error a cada iteración, y la zona sombreada representa

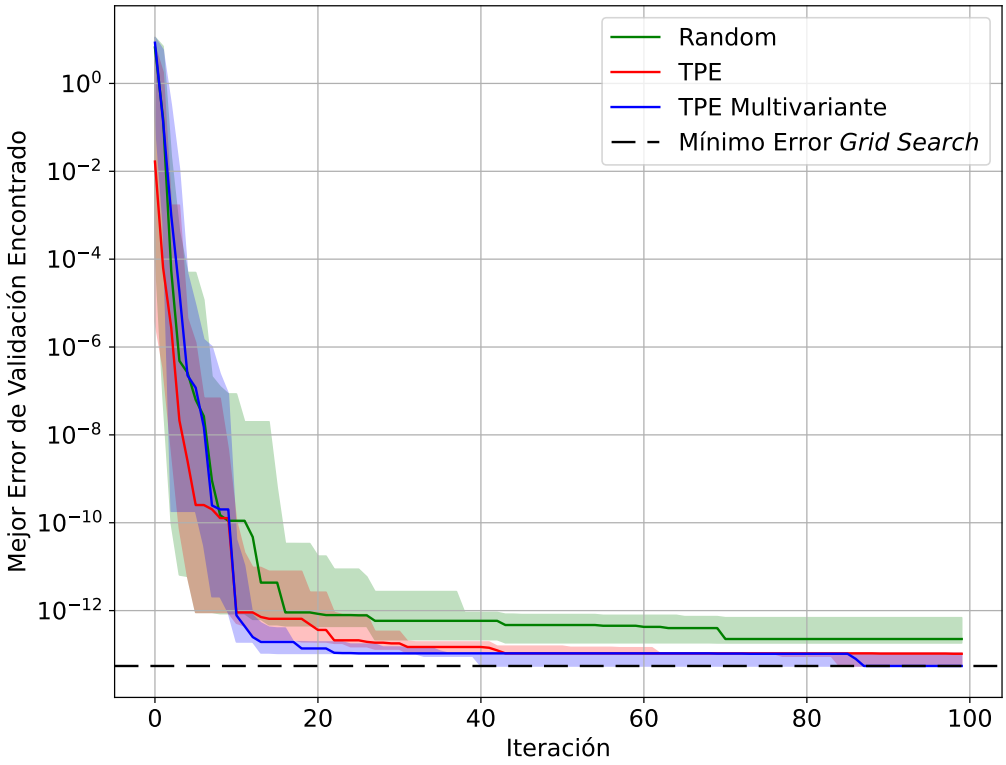

Figura 4.1: Comparación de convergencia. Se muestran los cuartiles para 20 optimizaciones realizadas, en cada caso.

Algoritmo	AUC (Mediana)	Mejor $Mediana(y)$ encontrada
Búsqueda Aleatoria	$2{,}59 \times 10^{-10}$	$2{,}26 \times 10^{-13}$
TPE	$1{,}20 \times 10^{-11}$	$1{,}04 \times 10^{-13}$
TPE Multivariante	$5{,}20 \times 10^{-12}$	$5{,}48 \times 10^{-14}$

Tabla 4.1: Comparación entre algoritmos de optimización.

los cuartiles. Aparte, en linea de trazo se marca el mejor error obtenido realizando una búsqueda exhaustiva (*grid search*).

En las primeras 10 repeticiones los tres métodos son equivalentes, pues para el algoritmo TPE (tanto el normal como el multivariable) se parte de un muestreo aleatorio de 10 observaciones. En la figura 4.1 se ve claramente que luego de las décima iteración se produce el cambio más notorio entre los tres métodos. Gráficamente se puede ver que ambas versiones del algoritmo TPE dan un mejor resultado que la búsqueda aleatoria, pero aparte de esto se pueden utilizar métricas como el **mejor valor encontrado** para la mediana de y o el **área bajo la curva** o **AUC** (*Area Under the Curve*)[33]. En la tabla 4.1 están los resultados de estas métricas, considerando solo las últimas 90 iteraciones.

Búsqueda Exhaustiva

La búsqueda exhaustiva o *grid search* consiste en probar todas las combinaciones posibles dentro de un espacio de hiperparámetros para seleccionar la solución óptima. Es decir que si se quiere buscar la combinación óptima de (n_{max}, l_{max}) para un rango de valores $n_{max} \in N$, $l_{max} \in L$ se deberán probar todas las combinaciones posibles del producto cartesiano $N \times L = \{(n_{max}, l_{max}) | n_{max} \in N, l_{max} \in L\}$.

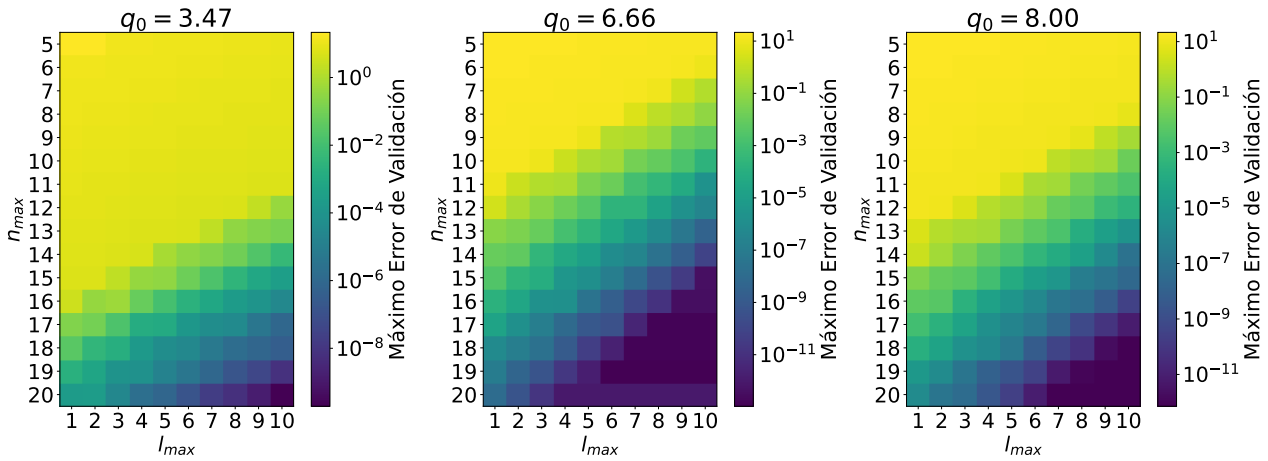

Figura 4.2: Máximo error de validación en función de n_{max} y l_{max} para tres diferentes semillas q_0.

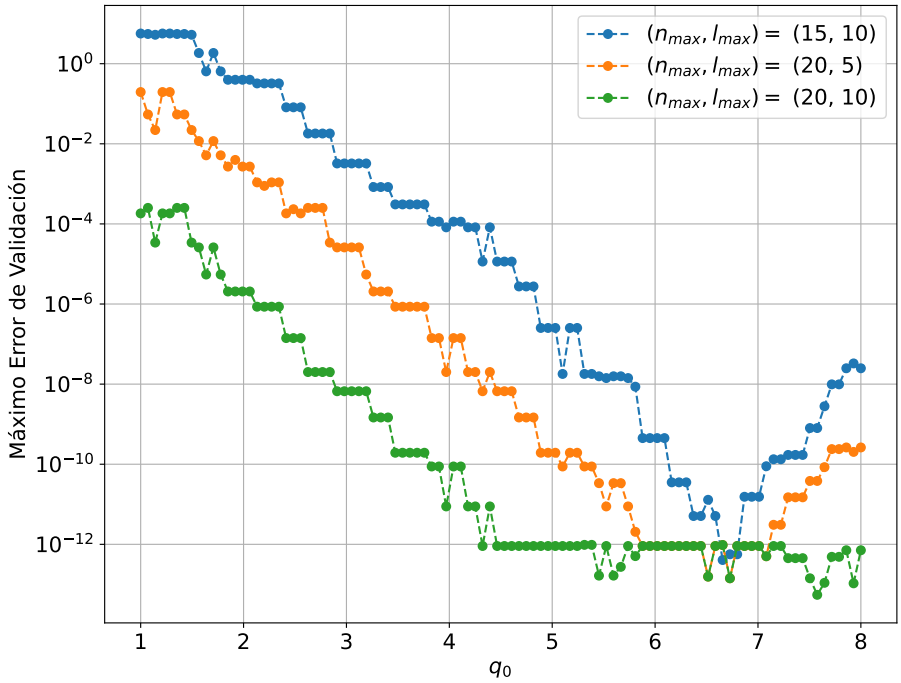

Figura 4.3: Máximo error de validación en función de la semilla q_0 para distintas combinaciones de (n_{max}, l_{max})

Si bien no se puede graficar el error en función de los tres hiperparámetros a la vez, se puede obtener bastante información al dejar fijo uno o dos hiperparámetros. Por

ejemplo en la figura 4.2 se observa el error de validación en función de las combinaciones posibles de n_{max} y l_{max} para tres diferentes semillas.

Luego en la figura 4.3 se ven los resultados de variar únicamente la semilla para diferentes combinaciones de n_{max} y l_{max}. En este conjunto de datos se observa que para $(n_{max}, l_{max}) = (20, 10)$ hay una diferencia de 9 ordenes de magnitud entre la peor y la mejor semilla (datos en color verde). Sin embargo para $(n_{max}, l_{max}) = (15, 10)$ la diferencia es de 13 ordenes de magnitud (datos de color azul). Además el valor óptimo de la semilla no coincide exactamente en estos dos ejemplos, aunque tengan un comportamiento similar. Es decir que la influencia de la semilla depende del resto de hiperparámetros, sobre todo se tiene que tener en cuenta que en este caso la tolerancia *greedy* tenía un valor $\varepsilon = 1 \cdot 10^{-12}$, por lo que no se va a obtener un resultado mucho mejor que este.

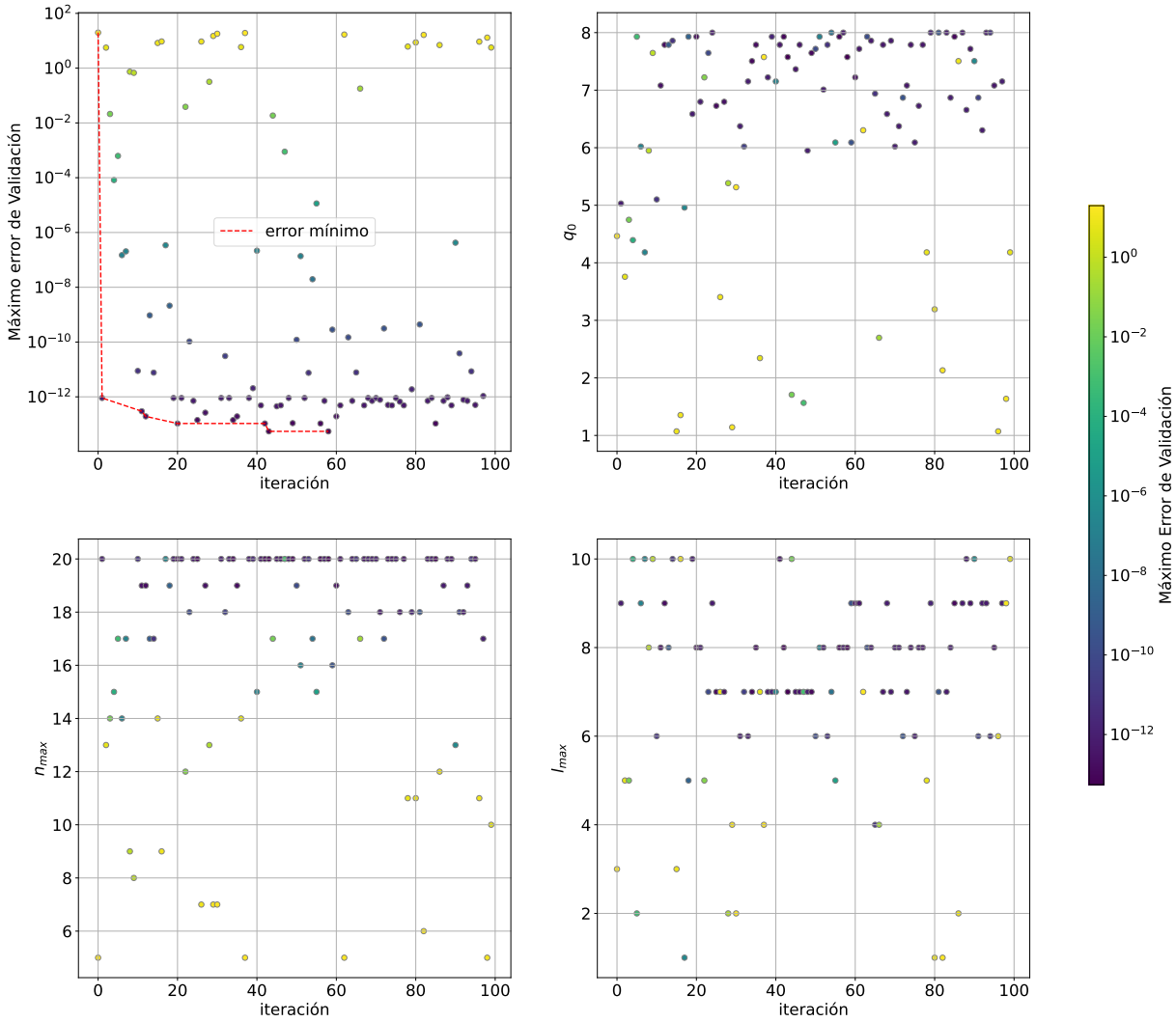

Figura 4.4: Optimización con 100 iteraciones utilizando el algoritmo TPE multivariante para el conjunto de entrenamiento con semilla unidimensional.

Tiempo de Optimización

Si bien la búsqueda exhaustiva garantiza encontrar el mejor resultado posible dentro del espacio de búsqueda, el tiempo necesario para realizar la búsqueda hace que el método no sea aplicable a casos relativamente complejos. En este caso sencillo, con 16000 combinaciones, la búsqueda requirió **25 horas** para completarse. En cambio las optimizaciones realizadas utilizando los algoritmos TPE tardaron una media de **8 minutos**.

Por último en la figura 4.4 se puede ver gráficamente el proceso de optimización utilizando el algoritmo TPE multivariable.

4.1.2. Optimización Completa

Utilizando un conjunto de entrenamiento con 70 valores discretos de q equidistantes en el rango [1, 8] y 20 de χ_z ($\chi_{z_1} = \chi_{z_2}$) en el rango [-0.8, 0.8], dando lugar a un total de 1400 funciones de onda, se muestran los resultados de optimizar el máximo error de validación utilizando un conjunto de validación con 100 valores para q y 30 vaores para χ_z con un total de 3000 funciones de onda.

La optimización se realizó en los siguientes espacios de búsqueda:

$$
\begin{aligned}
n_{max} &\in \{10, 11, 12, ..., 60\},\\
l_{max} &\in \{2, 3, 4, ..., 20\},\\
\varepsilon &\in \{10^{-20}, 10^{-19}, 10^{-18}, ..., 10^{-4}\},\\
Q_0 &= \{q_0 \mid q_0 = 1 + i\Delta q,\ i \in \mathbb{N} : 0 \leq i \leq 69,\ \Delta q = 7/69\},\\
X_0 &= \{\chi_{z_0} \mid \chi_{z_0} = -0{,}8 + j\Delta\chi_z,\ j \in \mathbb{N} : 0 \leq j \leq 19,\ \Delta\chi_z = 1{,}6/19\},\\
\hat{\Lambda}_0 &\in \{(q_0, \chi_{z_0}) \mid q_0 \in Q_0,\ \chi_{z_0} \in X_0\}.
\end{aligned}
$$

En total se realizaron 500 iteraciones, y el mejor máximo error de validación obtenido en la iteración número 269 fue de 1.45×10^{-6} con los hiperparámetros:

$$
\begin{aligned}
n^*_{max} &= 59,\\
l^*_{max} &= 4,\\
\varepsilon^* &= 10^{-17},\\
q^*_0 &= 7{,}899,\\
\chi^*_{z_0} &= 0{,}716.
\end{aligned}
$$

En la figura 4.5 se observa la evolución de la optimización realizada, que requirió

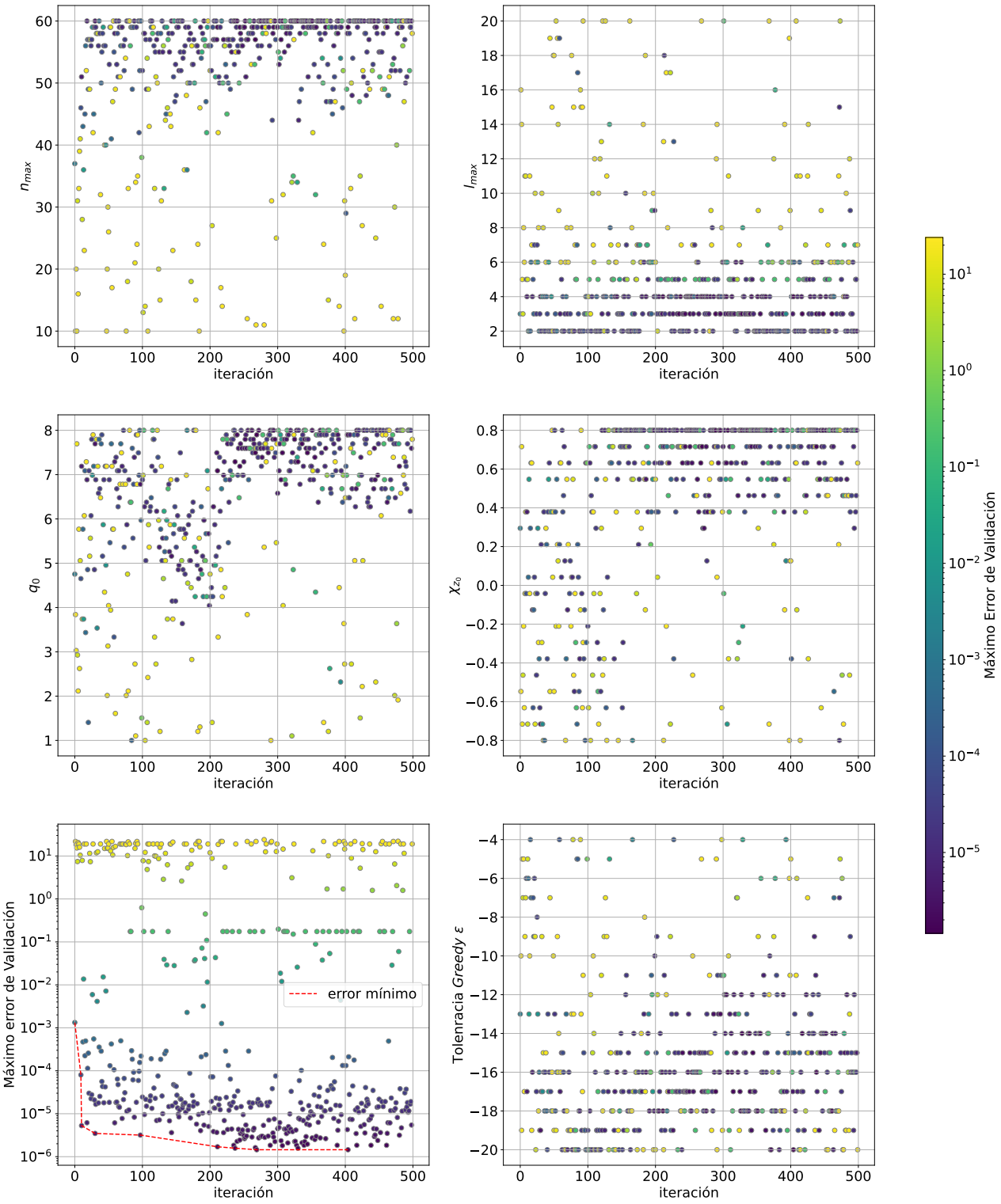

Figura 4.5: Optimización con 500 iteraciones para semilla de dos dimensiones utilizando el algoritmo TPE Multivariante.

alrededor de 8 horas para completarse. Se puede ver que para cada hiperparámetro se observa una convergencia a cierto valor, pero sin dejar de lado la exploración, es decir, que se siguen evaluando hiperparámetros fuera del rango que parece óptimo, de forma que se evita caer mucho tiempo en mínimos locales.

Importancia de los Hiperparámetros

Una vez realizada una optimización se puede estimar la importancia relativa de cada hiperparámetro con el algoritmo fANOVA [34]. Básicamente la idea es dividir la varianza total en distintos componentes que representen la varianza producida por cada hiperparámetro. En la figura 4.6 se observa a la izquierda en naranja los resultados para la optimización realizada, y a la derecha en azul se ven los resultados para otro espacio de búsqueda, esta vez con $n_{max} \in \{20, ..., 30\}$ y $l_{max} = \{2, ..., 8\}$ (es decir que se redujo el espacio de búsqueda para n_{max} y l_{max}). Se puede ver que la importancia que tienen los hiperparámetros depende claramente del espacio de búsqueda.

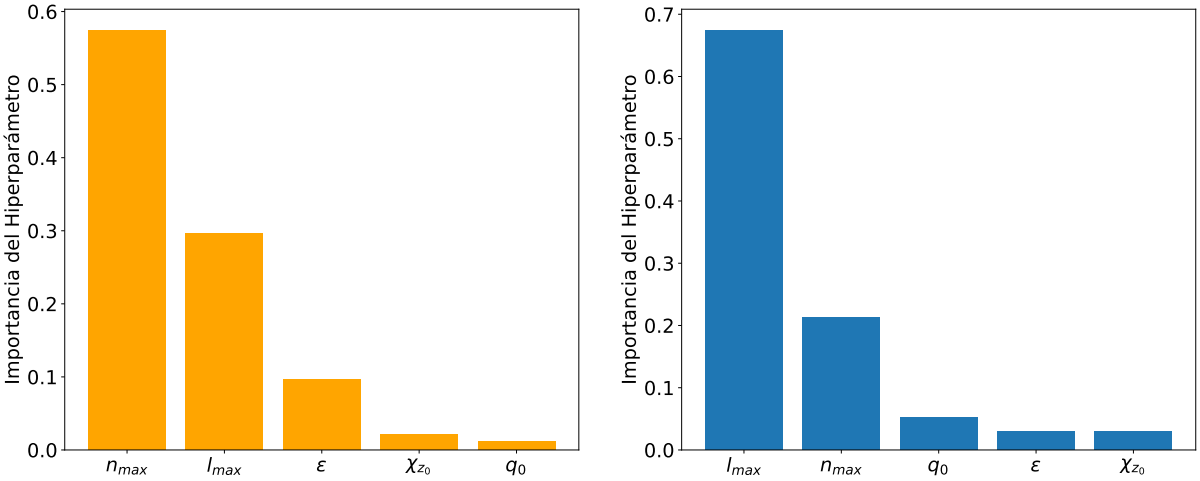

Figura 4.6: Importancia relativa de los hiperparámetros para dos espacios de búsqueda diferentes.

4.2. Optimización Multiobjetivo

En esta sección se encuentran los resultados más relevantes de la optimización multiobjetivo, que optimiza el máximo error de evaluación al mismo tiempo que el tiempo necesario para proyectar el conjunto de validación a la base creada.

4.2.1. Frente de Pareto

Utilizando un conjunto de entrenamiento con mil funciones de ondas equidistantes en el espacio del parámetro unidimensional $q : 1 < q < 8$, y un conjunto de validación

diez veces más denso (diez mil funciones de onda) se optimizó $\mathbf{x} = (n_{max}, l_{max}, \varepsilon, \hat{\Lambda}_0)$, en los siguientes intervalos:

$$
\begin{aligned}
n_{max} &\in \{10, 11, 12, ..., 20\}, \\
l_{max} &\in \{1, 2, 3, ..., 10\}, \\
\varepsilon &\in \{10^{-20}, 10^{-19}, 10^{-18}, ..., 10^{-6}\}, \\
\hat{\Lambda}_0 &\in \{q_0 \mid q_0 = 1 + i\Delta q,\ i \in \mathbb{N} : 0 \leq i \leq 999,\ \Delta q = 7/999\}.
\end{aligned}
$$

Una buena forma de visualizar los resultados es graficando ambos objetivos a la vez. En la figura 4.7 cada punto representa una observación, y lo interesante de este gráfico es que puede observar fácilmente el frente de Pareto (en color naranja). El frente de Pareto está conformado por aquellas observaciones que no sean dominadas por ninguna otra, pero son incomparables entre sí, por lo que este conjunto reemplaza a $\mathbf{x}^*$. Una vez obtenido este conjunto se debe elegir que objetivo es más importante y seleccionar una configuración $\mathbf{x}$ dentro del conjunto de Pareto.

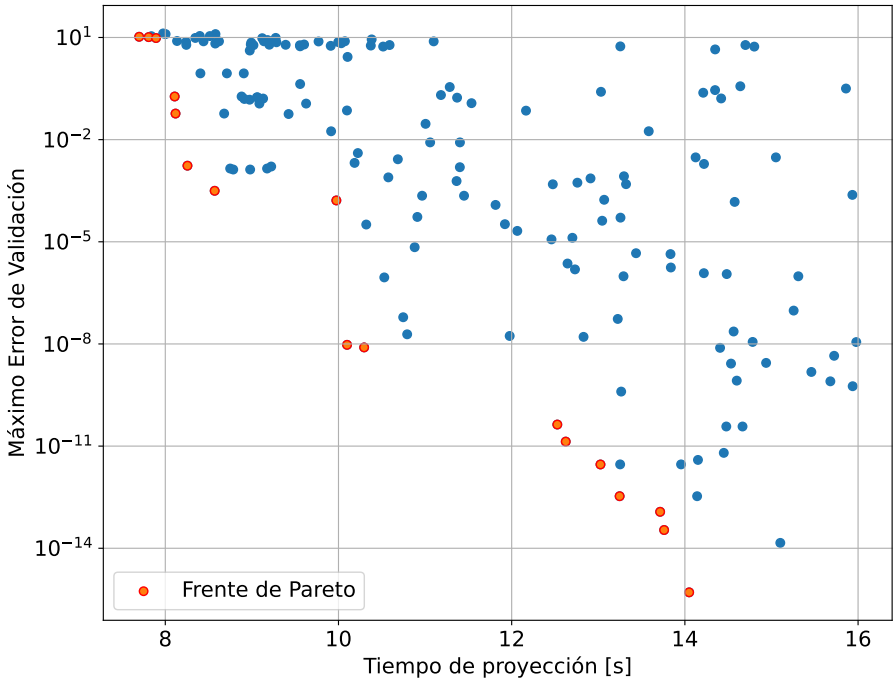

Figura 4.7: Máximo error de validación versus el tiempo de proytección. Cada punto representa una observación. En color naranja se marca el frente de Pareto.

4.2.2. Tiempo de Proyección versus Hiperparámetros

La optimización multiobjetivo resulta muy interesante, pero en el contexto de ondas gravitacionales, se observa que hay una gran dependencia entre el tiempo de proyección

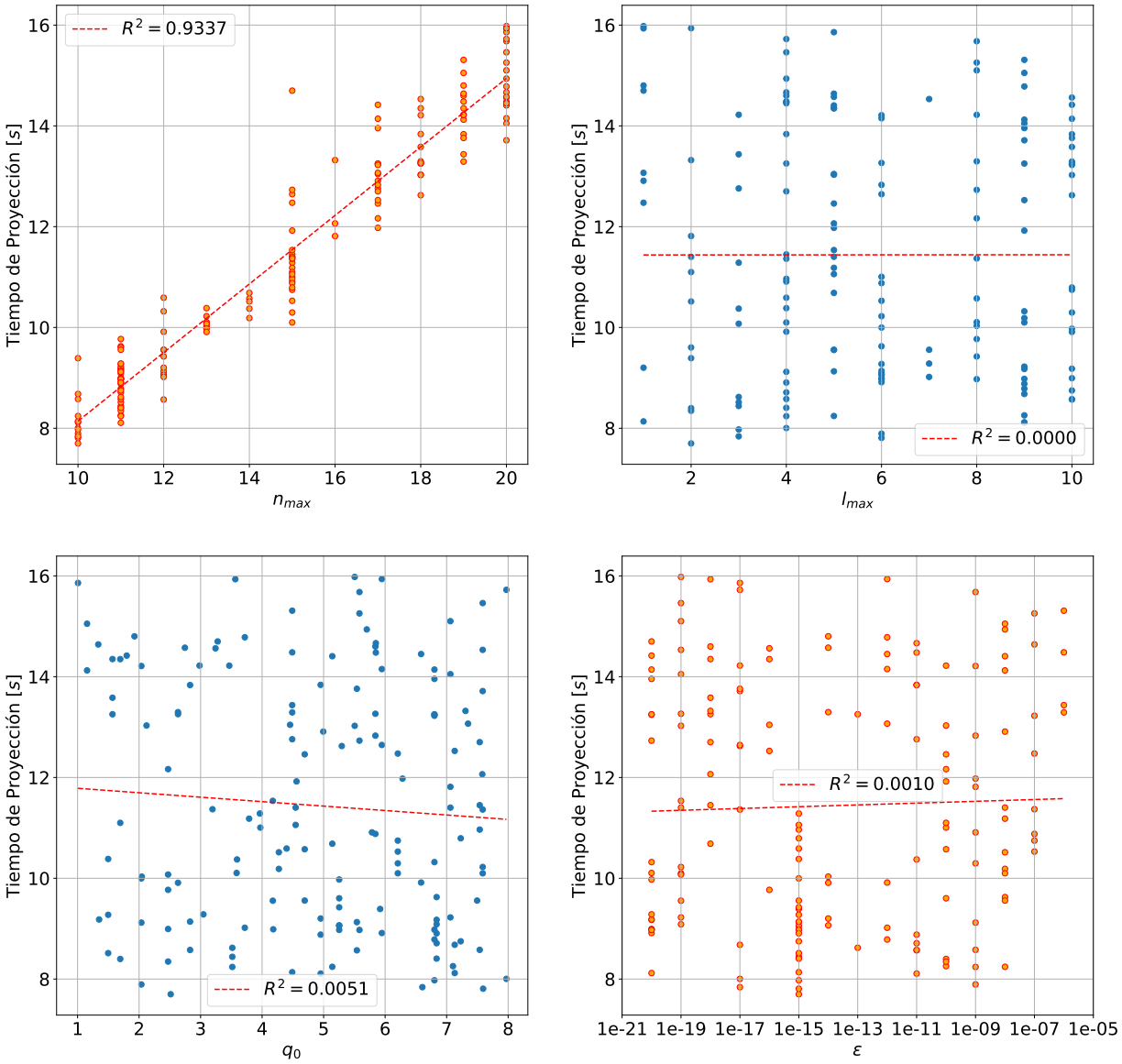

Figura 4.8: Relación entre el tiempo de ejecución y los diferentes hiperparámetros para una optimización de 160 iteraciones.

y n_{max}. Mas bien, como se observó en la sección de bases reducidas *hp greedy*, el tiempo de proyección depende casi exclusivamente del valor de n_{max}, siempre que el conjunto de entrenamiento sea lo suficientemente denso ($n_{max} \cdot 2^{l_{max}} < N$).

En la figura 4.8 se puede ver la dependencia entre los diferentes hiperparámetros y el tiempo de proyección en segundos, para las observaciones realizadas. Claramente hay una tendencia bastante lineal al considerar n_{max}.

Capítulo 5

Conclusiones

Acá van las conclusiones

Bibliografía

[1] Holst, M., Sarbach, O., Tiglio, M., Vallisneri, M. The emergence of gravitational wave science: 100 years of development of mathematical theory, detectors, numerical algorithms, and data analysis tools, 2016. 1

[2] Abbott, B. P., *et al.* Observation of Gravitational Waves from a Binary Black Hole Merger. *Phys. Rev. Lett.*, **116** (6), 061102, 2016. 1

[3] Veitch, J., Raymond, V., Farr, B., Farr, W., Graff, P., Vitale, S., *et al.* Parameter estimation for compact binaries with ground-based gravitational-wave observations using the LALInference software library. *Physical Review D*, **91** (4), feb 2015. 1

[4] Thrane, E., Talbot, C. An introduction to bayesian inference in gravitational-wave astronomy: Parameter estimation, model selection, and hierarchical models. *Publications of the Astronomical Society of Australia*, **36**, 2019. 1

[5] Field, S. E., Galley, C. R., Hesthaven, J. S., Kaye, J., Tiglio, M. Fast prediction and evaluation of gravitational waveforms using surrogate models. *Physical Review X*, **4** (3), jul 2014. 1

[6] Tiglio, M., Villanueva, A. Reduced order and surrogate models for gravitational waves. *Living Rev. Rel.*, **25** (1), 2, 2022. 2, 5, 7

[7] Hesthaven, J., Rozza, G., Stamm, B. Certified Reduced Basis Methods for Parametrized Partial Differential Equations. 2016. 2

[8] Chen, Y., Hesthaven, J. S., Maday, Y., Rodríguez, J. Certified reduced basis methods and output bounds for the harmonic maxwell's equations. *SIAM Journal on Scientific Computing*, **32** (2), 970–996, 2010. URL https://doi.org/10.1137/09075250X.

[9] Field, S. E., Galley, C. R., Herrmann, F., Hesthaven, J. S., Ochsner, E., Tiglio, M. Reduced basis catalogs for gravitational wave templates. *Phys. Rev. Lett.*, **106**, 221102, Jun 2011. URL https://link.aps.org/doi/10.1103/PhysRevLett.106.221102.

[10] Prud'homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., *et al.* Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods . *Journal of Fluids Engineering*, **124** (1), 70–80, 11 2001. URL https://doi.org/10.1115/1.1448332.

[11] Quarteroni, A., Manzoni, A., Negri, F. Reduced basis methods for partial differential equations: An introduction. 2015. 2

[12] Centrella, J., Baker, J. G., Kelly, B. J., van Meter, J. R. Black-hole binaries, gravitational waves, and numerical relativity. *Reviews of Modern Physics*, **82** (4), 3069–3119, nov 2010. 3

[13] Varma, V., Field, S. E., Scheel, M. A., Blackman, J., Kidder, L. E., Pfeiffer, H. P. Surrogate model of hybridized numerical relativity binary black hole waveforms. *Physical Review D*, **99** (6), mar 2019. 3

[14] Cerino, F., Diaz-Pace, J. A., Tiglio, M. An automated parameter domain decomposition approach for gravitational wave surrogates using hp-greedy refinement, 12 2022. 2, 5, 10, 12

[15] Pinkus, A. n-widths in approximation theory. 1985. 6

[16] Magaril-Il'yaev, G. G., Osipenko, K. Y., Tikhomirov, V. M. On exact values of n-widths in a hilbert space. *Journal of Approximation Theory*, **108** (1), 97–117, 2001. URL https://www.sciencedirect.com/science/article/pii/S002190450093497X. 7

[17] Binev, P., Cohen, A., Dahmen, W., DeVore, R., Petrova, G., Wojtaszczyk, P. Convergence rates for greedy algorithms in reduced basis methods. *SIAM Journal on Mathematical Analysis*, **43** (3), 1457–1472, 2011. URL https://doi.org/10.1137/100795772. 7

[18] Field, S. E., Galley, C. R., Hesthaven, J. S., Kaye, J., Tiglio, M. Fast prediction and evaluation of gravitational waveforms using surrogate models. *Phys. Rev. X*, **4**, 031006, Jul 2014. URL https://link.aps.org/doi/10.1103/PhysRevX.4.031006. 7

[19] Herrmann, F., Field, S. E., Galley, C. R., Ochsner, E., Tiglio, M. Towards beating the curse of dimensionality for gravitational waves using Reduced Basis. *Phys. Rev. D*, **86**, 084046, 2012. 7

[20] DeVore, R., Petrova, G., Wojtaszczyk, P. Greedy algorithms for reduced bases in banach spaces, 2012. URL https://arxiv.org/abs/1204.2290. 9

[21] Caudill, S., Field, S. E., Galley, C. R., Herrmann, F., Tiglio, M. Reduced basis representations of multi-mode black hole ringdown gravitational waves. *Classical and Quantum Gravity*, **29** (9), apr 2012. 9

[22] Hesthaven, J. S., Gottlieb, S., Gottlieb, D. Spectral Methods for Time-Dependent Problems. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2007. 10

[23] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., de Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, **104** (1), 148–175, 2016. 2, 19

[24] Brochu, E., Cora, V. M., de Freitas, N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010. URL https://arxiv.org/abs/1012.2599. 2, 19, 22

[25] Dewancker, I., McCourt, M., Clark, S. Bayesian optimization primer, 2015. URL https://app.sigopt.com/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf. 19

[26] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B. Algorithms for hyper-parameter optimization. En: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (eds.) Advances in Neural Information Processing Systems, tomo 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf. 19, 23

[27] Feurer, M., Hutter, F. Hyperparameter Optimization, págs. 3–33. Cham: Springer International Publishing, 2019. URL https://doi.org/10.1007/978-3-030-05318-5_1. 20

[28] Jones, D. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, **21**, 345–383, 12 2001. 22

[29] Bergstra, J., Yamins, D., Cox, D. D. Making a science of model search, 2012. URL https://arxiv.org/abs/1209.5111. 23

[30] Ozaki, Y., Tanigaki, Y., Watanabe, S., Onishi, M. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems, 2020. URL https://doi.org/10.1145/3377930.3389817. 23, 24

[31] Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M. Optuna: A next-generation hyperparameter optimization framework. En: Proceedings of the 25rd ACM

SIGKDD International Conference on Knowledge Discovery and Data Mining. 2019. 24, 27

[32] Ozaki, Y., Tanigaki, Y., Watanabe, S., Nomura, M., Onishi, M. Multiobjective tree-structured parzen estimator. *J. Artif. Int. Res.*, **73**, may 2022. URL https://doi.org/10.1613/jair.1.13188. 24, 26

[33] Dewancker, I., McCourt, M. J., Clark, S. C., Hayes, P., Johnson, A., Ke, G. A strategy for ranking optimization methods using multiple criteria. En: AutoML@ICML. 2016. 28

[34] Hutter, F., Hoos, H., Leyton-Brown, K. An efficient approach for assessing hyperparameter importance. En: E. P. Xing, T. Jebara (eds.) Proceedings of the 31st International Conference on Machine Learning, tomo 32 de *Proceedings of Machine Learning Research*, págs. 754–762. Bejing, China: PMLR, 2014. URL https://proceedings.mlr.press/v32/hutter14.html. 33

Publicaciones asociadas

1. Mi primer aviso en la revista **ABC**, 1996
2. Mi segunda publicación en la revista **ABC**, 1997

Agradecimientos

A todos los que se lo merecen, por merecerlo