TESIS DE LICENCIATURA EN FÍSICA

OPTIMIZACIÓN DE HIPERPARÁMETROS EN MODELOS REDUCIDOS ADAPTATIVOS, CON APLICACIONES A ONDAS GRAVITACIONALES Y EL PROYECTO LIGO

Atuel E. Villegas A.

Dr. Manuel Tiglio
Director

Dr. Carlos Figueroa
Co-director

Miembros del Jurado
Dra. Ana Georgina Elias
Dra. Graciela Molina
Dr. Benjamín Straube

5 de Mayo de 2023

Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán
Argentina

A mi familia
A mis amigos
A mis compañeros´

Índice de contenidos

Resumen

La inferencia de parámetros para una colisión binaria de agujeros negros es un área de gran importancia dentro de la ciencia de ondas gravitacionales, sobre todo con el trabajo conjunto de los interferómetros LIGO, VIRGO y KAGRA generando una gran cantidad de datos para ser analizados. El estándar actual para realizar esta inferencia requiere la producción de funciones de onda en tiempo real, lo que no es posible utilizando relatividad numérica. Los modelos sustitutos de orden reducido son una gran alternativa que permite generar resultados precisos en el orden de los milisegundos.

En esta tesis se trató el problema de la optimización de hiperparámetros para un sistema de aprendizaje supervisado, el cual consiste en la primera etapa de la construcción de un modelo sustituto. Este sistema de aprendizaje es un refinamiento del método de las bases reducidas, ya utilizado en la construcción de modelos sustitutos.

Para la optimización se utilizaron métodos bayesianos, los cuales se vienen utilizando bastante dentro de la ciencia de datos en los últimos años debido a la necesidad de crear modelos cada vez más complejos y precisos. Esta optimización se realizó utilizando la librería llamada Optuna escrita en Python. Se comparó el método con la búsqueda aleatoria y se obtuvieron buenos resultados, mostrando una clara superioridad de la optimización bayesiana, por lo menos en este contexto.

Palabras clave: ONDAS GRAVITACIONALES, BASES REDUCIDAS, OPTIMIZACIÓN DE HIPERPARÁMETROS

Abstract

Parameter inference for binary black hole collisions is an area of great importance within gravitational wave science, especially with the joint work of the LIGO, VIRGO and KAGRA interferometers generating a large amount of data to be analyzed. The current standard for performing this inference requires the production of wave functions in real time, which is not possible using numerical relativity methods. Reduced order surrogate models are a great alternative that allows the generation of accurate results in the order of milliseconds.

In this thesis the problem of hyperparameter optimization for a supervised learning system is treated, which consists of the first stage of the construction of a surrogate model. This learning system is a refinement of the reduced basis method, already used in the construction of surrogate models.

Bayesian methods were used for the optimization, which have been widely used in data science in recent years due to the need to create increasingly complex and accurate models. This optimization was performed using the Optuna library written in Python. The method was compared with random search and good results were obtained, showing a clear superiority of Bayesian optimization, at least in this context.

Keywords: GRAVITATIONAL WAVES, REDUCED BASIS, HYPERPARAMETER OPTIMIZATION

Capítulo 1

Introducción

La ciencia de ondas gravitacionales es un área que ha experimentado un rápido crecimiento en las últimas décadas. Este crecimiento llevó, en septiembre de 2015, a la primera detección obtenida de ondas gravitacionales producto de una colisión binaria de agujeros negros [1, 2]. Al momento de escribir esta tesis (últimos meses de 2022), los interferómetros 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 solo confirma una de las predicciones más relevantes de las ecuaciones de Einstein, sino que también aporta una gran cantidad de información sobre 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 espines) 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 resolver las ecuaciones de Einstein; ya que utilizando relatividad numérica se necesitan meses de cómputo para obtener la función de onda producto de una colisión binaria. 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 tesis se trabajó dentro del marco de los modelos sustitutos de orden reducido [5, 6], una alternativa que logra generar soluciones de alta precisión, comparables a los métodos de relatividad numérica, pero en tiempo real y en una simple computadora portátil. Dentro de este marco se utilizó el enfoque de las bases reducidas [7–11], un método espectral que permite representar un espacio de soluciones a partir de sus n elementos más representativos, de forma que se elimina la información redundante a la hora de representar dicho espacio.

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

Objetivo

En este trabajo se expande el método de bases reducidas con el refinamiento *hp-greedy* [12], el cual divide iterativamente el espacio de parámetros en distintos subdominios, construyendo bases reducidas locales en lugar de una global, y dando como resultado una estructura de árbol binario.

La ventaja de este refinamiento está en que las bases locales tienen menor dimensión que la base global, sin que esto implique una pérdida de precisión. Esto se traduce en un modelo más rápido y eficiente de evaluar, pues en lugar de representar una solución proyectándola a la base global, se la proyectará a una de las bases locales (la que corresponda según los parámetros de la solución). Pero junto a esta ventaja aparece la complejidad añadida por los hiperparámetros, los cuales son parámetros que que gobiernan la construcción de la base *hp-greedy* (por ejemplo, la profundidad máxima del árbol).

El objetivo de esta tesis es encontrar la configuración de hiperparámetros que de lugar a una base óptima. Partiendo de un espacio de hiperparámetros, una base reducida *hp-greedy* óptima será aquella con la configuración de hiperparámetros que de lugar al menor error de representación a la hora de proyectar un dado espacio.

Metodología: Optimización Bayesiana

Una base reducida *hp-greedy* constituye un sistema de aprendizaje supervisado con numerosas configuraciones de hiperparámetros, cuyo ajuste afecta significativamente el rendimiento del sistema. Por lo tanto, resulta fundamental optimizar la construcción de este sistema.

La optimización bayesiana [13, 14] se ha convertido en un método de optimización cada vez más utilizado en diferentes áreas de la ciencia de datos, con resultados sobresalientes en el entrenamiento de diversos modelos de aprendizaje automático y aprendizaje profundo (no confundir con la inferencia bayesiana, utilizada para la estimación de parámetros). La elección manual de los hiperparámetros de una base reducida *hp-greedy* suele generar resultados mediocres, por lo que es necesario realizar múltiples pruebas para obtener mejoras significativas en el resultado inicial. La optimización bayesiana automatiza este proceso y además cada nueva evaluación tiene en cuenta las evaluaciones realizadas previamente.

Particularmente en este trabajo se utilizó el estimador de Parzen con estructura arbórea o *Tree-Parzen Estimator* (TPE), una estrategia popular dentro de la optimización bayesiana, cuyo algoritmo está implementado en la librería Optuna [15] [2].

Los resultados obtenidos fueron favorables, mostrando una clara ventaja de la optimización bayesiana sobre una elección aleatoria de hiperparámetros.

[2] Ver https://optuna.org/

Optimización Multiobjetivo

Adicionalmente se utilizó un procedimiento que optimizó el tiempo requerido para realizar una proyección a la base *hp-greedy* final (al mismo tiempo que se optimizaba el error de representación).

El tiempo de proyección es el tiempo que tarda en proyectarse una función (o conjunto de funciones) al espacio generado por la base construida. Este segundo objetivo es relevante ya que se espera que un menor tiempo de proyección de lugar a un modelo más rápido de evaluar.

1.1. Representación de Ondas Gravitacionales

La colisión de dos agujeros negros es uno de los eventos que más energía libera en el universo. Una gran parte de esta energía se libera en forma de ondas gravitacionales [16]. Estas ondas, al igual que las ondas electromagnéticas, son transversales y viajan a la velocidad de la luz.

Para la representación de una onda gravitacional se utiliza la magnitud h, denominada *strain*, que es proporcional a la deformación relativa del espacio. Existen tres partes que pueden ser identificadas en la función de onda: *inspiral*, *merger* y *ringdown*.

En la primera etapa, (*inspiral*), los dos agujeros negros se orbitan mutuamente durante mucho tiempo, acercándose poco a poco a la vez que sus velocidades aumentan. Eventualmente los agujeros negros se acercan demasiado y se produce la colisión (*merger*), caracterizada por un solo horizonte de eventos. Es aquí dónde se produce el pico de intensidad de la función de onda. Por último, la amplitud disminuye rápidamente con forma similar a una oscilación amortiguada, denotando el final de la colisión (*ringdown*).

Una forma conveniente de expresar una onda gravitacional es a partir de una expansión en los armónicos esféricos con peso en espín $s = -2$, denotados por ${}_{-2}Y^{lm}(\theta, \phi)$:

$$h(\lambda; t, r, \theta, \phi) = \sum_{l=2}^{\infty} \sum_{m=-l}^{l} {}_{-2}Y^{lm}(\theta, \phi) h^{lm}(\lambda; t, r). \tag{1.1}$$

Para grandes distancias se puede expresar:

$$h(\lambda; t, r, \theta, \phi) \approx \frac{1}{r} \sum_{l=2}^{\infty} \sum_{m=-l}^{l} {}_{-2}Y^{lm}(\theta, \phi) h^{lm}(\lambda; t) \tag{1.2}$$

donde para cada par (l, m), $h(\lambda; t)$ es un valor complejo que denota dos tipos de polarizaciones; + y $\times$:

$$h(\lambda;t) = h_+(\lambda;t) + ih_\times(\lambda;t) \tag{1.3}$$

La polarización + representa una onda que estira a lo largo de un eje y comprime a lo largo del otro, mientras que la polarización × hace lo mismo pero en ejes rotados 45°.

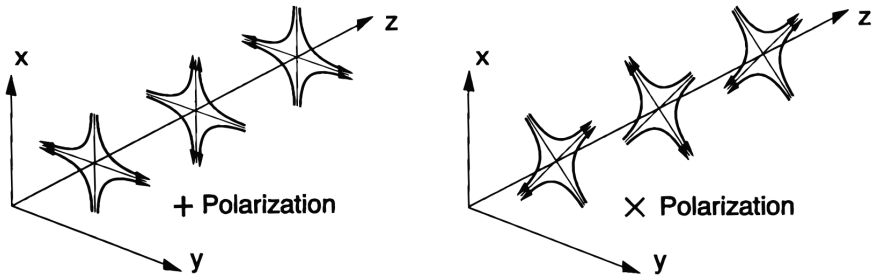

Figura 1.1: Esquema de lineas de fuerza para las polarizaciones + y × de una onda que se propaga en la dirección z [16].

El término λ se refiere al conjunto de parámetros que define a la onda producida por la colisión binaria (por ejemplo las masas o spines de los agujeros negros).

Datos Utilizados

Para esta tesis se utilizaron ondas gravitacionales generadas a partir del modelo híbrido *NRHybSur3dq8*[17] que utiliza relatividad numérica y aproximaciones post Newtonianas para colisiones de agujeros negros binarios.

Cada solución, es decir, cada onda $h := h(t;\lambda) := h_\lambda(t) := h_\lambda$ generada, parametrizada por λ, se representa con una serie temporal compleja de la forma (ver figura 1.2)

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

En el desarrollo de este trabajo λ tendrá 3 dimensiones relevantes, acotadas de la siguiente manera:

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

Además para tener un espacio de parámetros más sencillo, en todos los casos se trabajó con $\chi_{1_Z} = \chi_{2_Z}$, por lo que se puede hablar directamente del espín $\chi_Z = \chi_{1_Z} = \chi_{1_Z}$. También es importante mencionar que se trabajó únicamente con el modo

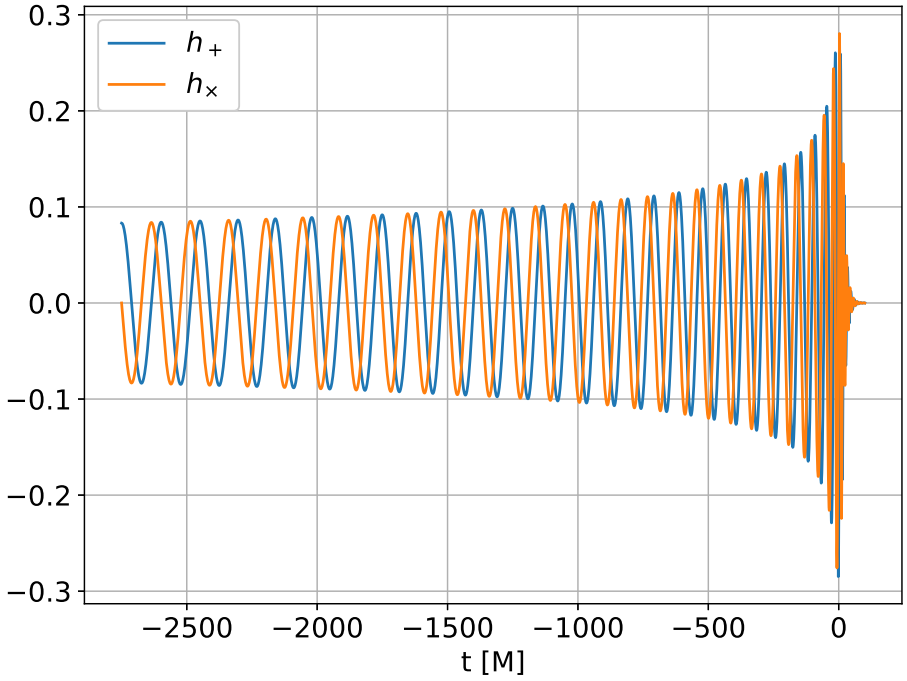

Figura 1.2: polarizaciones h_+ y $h_\times$ para $q = 3$, $\chi_{1z} = \chi_{2z} = 0$, en el modo $l = 2$, $m = 2$.

$(l, m) = (2, 2)$, pero los resultados obtenidos se pueden aplicar a todos los modos por igual (recordar ecuación (1.2)).

Se expresará un conjunto de N soluciones con $\mathcal{K}$, de la siguiente forma:

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

Este conjunto genérico puede entenderse como un muestreo de soluciones en el espacio el espacio de parámetros λ, donde cada elemento es una onda gravitacional.

y debido a que cada h_{λ_i} es una serie temporal, $\mathcal{K}$ se puede representar 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.

Conjuntos de Entrenamiento y Validación

Siguiendo el lenguaje utilizado en aprendizaje automático, se utilizará el concepto de conjunto de entrenamiento y conjunto de validación.

Un conjunto de entrenamiento es aquel que se utiliza para entrenar un algoritmo o modelo, mientras que el de conjunto de validación se utiliza luego del entrenamiento para asignar un puntaje al modelo entrenado.

Un conjunto de entrenamiento (o validación) consistirá en el conjunto que contenga pares $(\lambda_i, h_{\lambda_i})$, es decir que cada solución h_{λ_i} estará acompañada de los parámetros λ_i que la generaron:

$$\mathcal{T} := \{\lambda_i, h_{\lambda_i}\}_{i=1}^{N}$$

1.2. Modelos Sustitutos en Ciencia de Ondas Gravitacionales

Antes de continuar, en esta sección se explica muy brevemente la importancia del uso de modelos sustitutos en la inferencia de parámetros. Como se mencionó anteriormente, la inferencia (o estimación) de parámetros se realiza utilizando un método llamado **inferencia bayesiana** (no confundir con optimización bayesiana). A muy grandes rasgos se puede pensar en un proceso como el que se ejemplifica en la figura 1.3, donde la inferencia bayesiana tiene como entrada los resultados de una medición (por ejemplo realizada por el observatorio LIGO), y como salida devuelve el conjunto de parámetros λ_x que caracteriza a la onda medida.

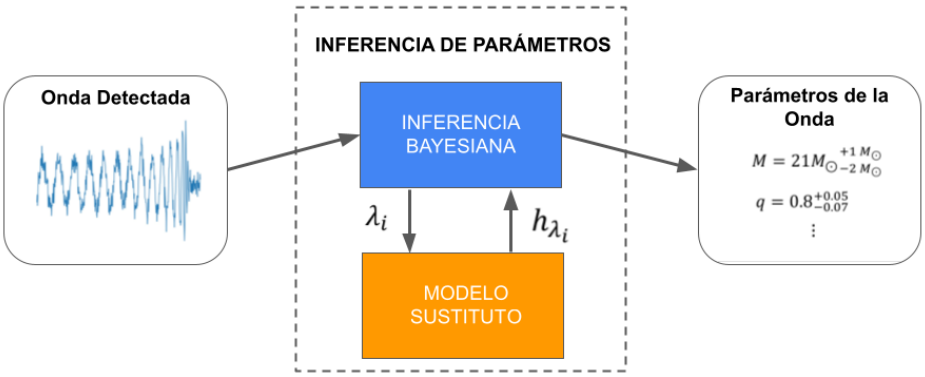

Figura 1.3: Esquema muy simplificado del proceso de inferencia de parámetros utilizando inferencia bayesiana y modelos sustitutos. El método de inferencia bayesiana necesita generar múltiples soluciones h_{λ_i} para diferentes parámetros λ_i. Estas soluciones son generadas por el modelo sustituto, el cual es capaz de realizar esta tarea en tiempo real.

El problema es que la inferencia bayesiana necesita generar múltiples soluciones en tiempo real, lo que no se puede lograr utilizando métodos de relatividad numérica,

que tardan meses en generar una solución. Aquí aparecen los modelos sustitutos, que sustituyen a la relatividad numérica y generan soluciones h_{λ_i} para cada conjunto de parámetros λ_i requerido para el proceso de inferencia bayesiana.

En esta tesis, como ya se mencionó, se trabaja en optimizar hiperparámetros para la construcción de bases reducidas, que conformarán el núcleo de un modelo sustituto.

Capítulo 2

Teoría de Aproximaciones: Bases Reducidas y Aprendizaje

En la primera sección de este capítulo se dará una introducción al enfoque de bases reducidas, un método de aproximación fundamental en el modelado de orden reducido [6].

Luego, en la segunda sección se ampliará con el refinamiento hp *greedy* [12]; una metodología que descompone de forma iterativa el dominio de parámetros para crear bases reducidas locales de forma adaptativa.

2.1. Bases Reducidas

El objetivo de este método es obtener una base que represente un espacio de soluciones de forma aproximada pero con alta precisión, evitando tener que resolver el problema completo múltiples veces para generar más soluciones.

Las bases reducidas son un método de expansión espectral, así como la expansión de Fourier o los polinomios 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 lo otros métodos mencionados).

Para la construcción de una base reducida se parte de un espacio de soluciones $\mathcal{F} := \{h_\lambda = h_\lambda(t) = h(\lambda, t)\}$, donde h_λ representa una onda gravitacional parametrizada por $\lambda \in \Omega$, siendo Ω un dominio compacto de parámetros. Cada parámetro λ es multidimensional y representa los parámetros mencionados anteriormente, como la relación entre masas o los espines de los agujeros negros. En la Figura 2.1 a) se representa un espacio Ω de dos dimensiones.

Luego se realiza un muestreo de $\mathcal{F}$ y Ω para construir un conjunto de entrenamiento $\mathcal{T} := \{\lambda_i, h_{\lambda_i}\}_{i=1}^{N}$ con N elementos. En la Figura 2.1 b) se representa un muestreo para un Ω de dos dimensiones, donde cada punto λ_i representa a su vez una función $h_{\lambda_i} \in \mathcal{F}$.

De este conjunto $\mathcal{T}$ se seleccionarán los elementos más representativos para construir una base $\{e_i\}_{i=1}^n$, generalmente con $n \ll N$, de forma que se pueda generar el conjunto $\{h_{\lambda_i}\}_i^N$, e incluso el espacio $\mathcal{F}$, con la combinación lineal:

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

Luego utilizando un conjunto de validación se puede obtener una medida del error o precisión con la que dicha base representa al espacio original.

2.1.1. Elección de una base óptima

La pregunta de que tan bien se puede aproximar el espacio $\mathcal{F}$ a partir de una base $\{e_i\}_{i=1}^n$ se puede caracterizar utilizando la definición de la distancia de Kolmogorov [18]:

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

Esta distancia es 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 $\|\cdot\|$ en (2.2) 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, se explica el significado de los mín, máx, 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:

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

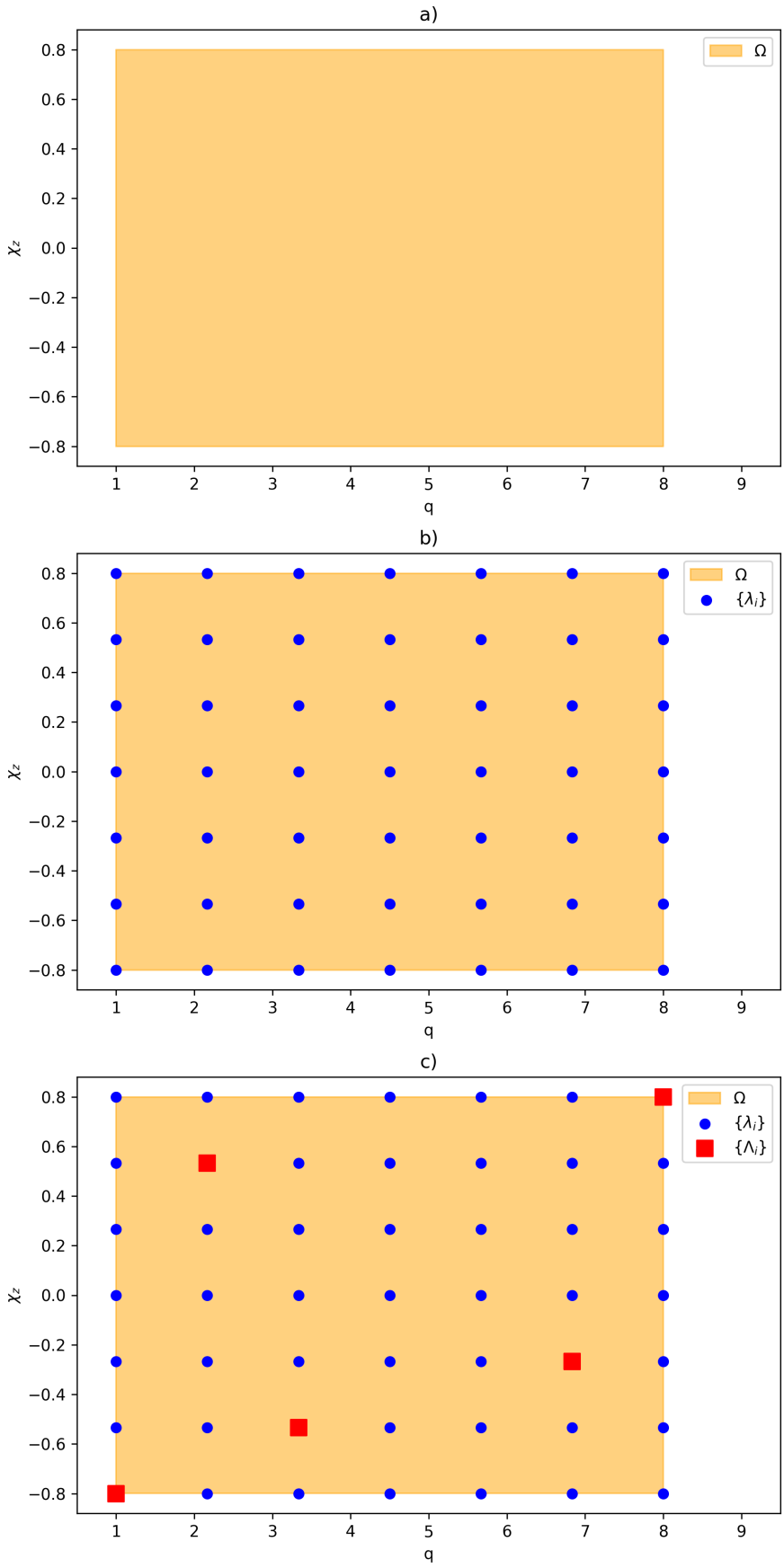

Figura 2.1: a) Representación del espacio Ω. b) Los puntos azules son resultado de un muestro equiespaciado en el espacio Ω, cada punto es un parámetro λ_i que representa a una onda h_{λ_i}. c) Los cuadrados rojos representan a los parámetros *greedy* $\{\Lambda_i\}$, a partir de los cuales se construye la base reducida.

con

$$\mathcal{P}_n h_\lambda = \sum_{i=1}^{n} c_{i,\lambda} e_i(t) \tag{2.4}$$

- $\text{máx}_{\lambda\in\Omega}$ hace referencia a que se tiene en cuenta el peor error (el máximo) en el espacio de parámetros Ω.

- Por último $\text{mín}_{\{e_i\}_{i=1}^n}$ 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 [19]. 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^∞ [20]. En el contexto de ondas gravitacionales se espera un comportamiento asintótico de convergencia exponencial con n [21, 22].

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 a la base el elemento que daba lugar al peor (máximo) error de representación con la base anterior. Es decir que en cada iteración se reducirá el máximo error de representación de todo el conjunto de entrenamiento.

Para cada solución h_λ del conjunto de entrenamiento, el error de representación es $\|h_\lambda - \mathcal{P}_n h_\lambda\|^2$, por lo tanto el máximo error de representación o "Error *greedy*", se define como :

$$\sigma_n := \max_{\lambda\in\{\lambda_i\}_{i=1}^N} \|h_\lambda - \mathcal{P}_n h_\lambda\|^2. \tag{2.5}$$

Este error es el *error de entrenamiento*, que disminuirá con cada iteración del algoritmo. También se hablará del *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 tres elementos:
  - 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},
  - El 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*. En la figura 2.1 c) se observan en color rojo 5 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.

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

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

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

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

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

se mencionó anteriormente, la medida de Kolmogorov d_n puede utilizarse como cota superior. Resultados obtenidos en [23] 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 2.2 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 [24]. Lo más relevante es que se puede 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.

2.2. Bases Reducidas hp Greedy

El nombre del método *hp-greedy* [12] surge de la combinación del *"refinamiento p"* y del *"refinamiento h"*. El *refinamiento p* proviene de los métodos espectrales con bases polinomiales [25] 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

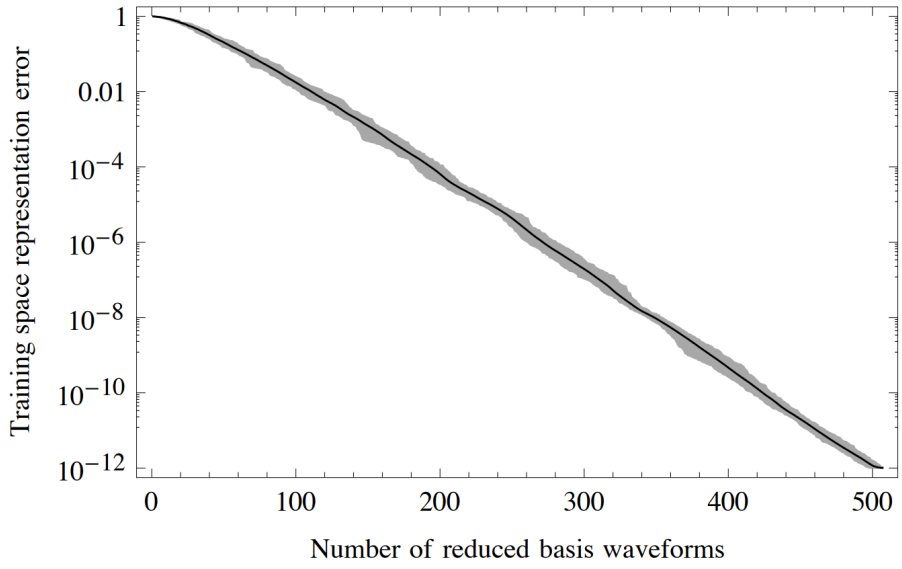

Figura 2.2: Error de representación en función del número de funciones de onda en la base reducida para un modo cuasinormal (QNM). Se probaron todas las semillas posibles del conjunto de entrenamiento; en negrita se ve el promedio, y el área sombreada representa los valores extremos [24].

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, y este se refina para obtener un mejor resultado. En el caso de las bases reducidas *hp-greedy* este último 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.
- Ω: espacio total de parámetros tal que $\Omega = \cup V$.
- V_1, V_2 : particiones de V.
- Λ_V : parámetros *greedy* en 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 $V \subseteq \Omega$ en dos subdominios V_1 y V_2. La división de cada dominio V se realizará en función de dos puntos de anclaje $\hat{\Lambda}_{V_1}$ y $\hat{\Lambda}_{V_2}$. Estos puntos de anclaje no son más que los primeros dos elementos del conjunto de parámetros *greedy* Λ_V. Como resultado de estas divisiones iterativas 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 próximos a $\hat{\Lambda}_{V_1}$, y λ_{V_2} con los λ_i más próximos 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: $\lambda_V, \hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2}$

```
λ_{V_1} = λ_{V_2} = ∅
for each λ_i ∈ λ_V do
    if d(λ_i, Λ̂_{V_1}) < d(λ_i, Λ̂_{V_2}) then
        λ_{V_1} = λ_{V_1} ∪ λ_i
    else if d(λ_i, Λ̂_{V_1}) > d(λ_i, Λ̂_{V_2}) then
        λ_{V_2} = λ_{V_2} ∪ λ_i
    else
        λ_{V'} = random choice([λ_{V_1}, λ_{V_2}])
        λ_{V'} = λ_{V'} ∪ λ_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:

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

y en general:

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

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

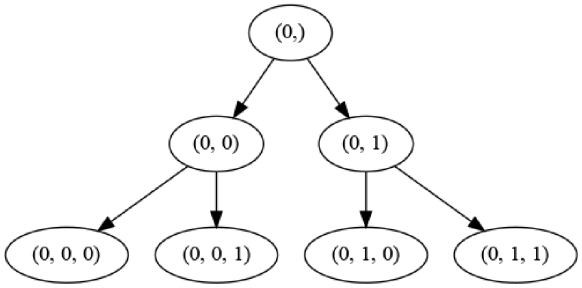

Figura 2.3: Representación de los nodos de un árbol con $l_{max} = 2$ [12].

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 $\sigma \leq \varepsilon$.

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. En la figura 2.4 se puede ver un esquema de la división de parámetros para el caso de un λ de una sola dimensión. Luego en la figura 2.5 se esquematiza el proceso de proyección de una función h_{λ_i} a la base creada.

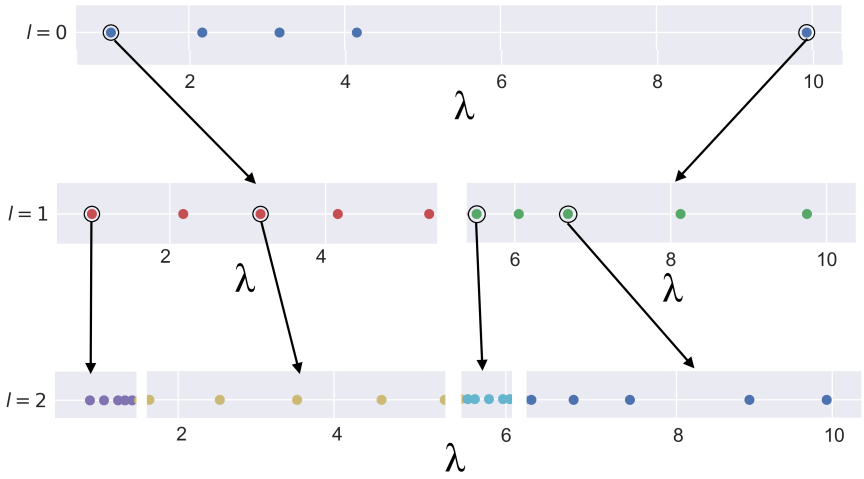

Figura 2.4: Ejemplo de partición del dominio de parametros para un parámetro λ unidimensional. Los puntos de colores representan a los parámetros *greedy* Λ_V para cada subdominio V, resaltando a los puntos de anclaje Λ_V con un círculo externo. Este diagrama muestra como se divide cada subdominio en función de sus puntos de anclaje. En este ejemplo $n_{max} = 5$ y $l_{max} = 2$.

2.2.3. Tiempo de Proyección

Un aspecto muy importante del refinamiento *hp-greedy*, que se mencionará seguido de ahora en adelante, es el tiempo de proyección a la base.

Al momento de proyectar una función h_{λ_i} a una base *hp-greedy*, primero se debe obtener la base local del nodo hoja (ver fig. 2.5) y luego se realiza la proyección a la base encontrada. Proyectar un conjunto de validación implica repetir este mismo proceso para todas las funciones del conjunto. Este proceso se realizará a lo largo de un tiempo t, que se denomina *tiempo de proyección*.

En el siguiente apartado se analiza el tiempo de proyección en diferentes bases *hp-greedy* en el contexto de ondas gravitacionales.

2.2.4. Aplicación a Ondas Gravitacionales

En esta subsección se utiliza un conjunto de ondas gravitacionales con parámetro bidimensional $\lambda = (q, \chi_z)$, donde $\chi_{1_z} = \chi_{2_z} = \chi_z$ es el espín de los agujeros negros y q es la relación entre sus masas. De esta forma se puede graficar fácilmente el dominio de parámetros.

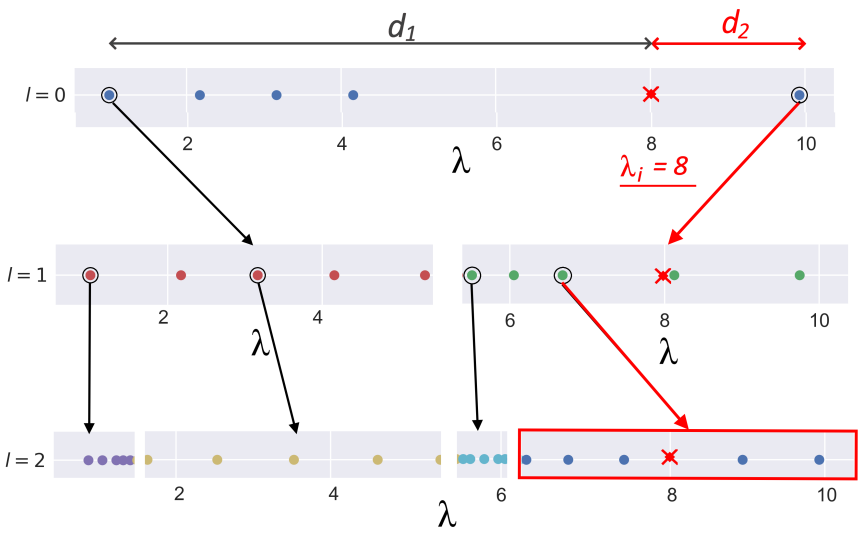

Figura 2.5: Proyección de una onda h_{λ_i} sobre la base de ejemplo de la figura 2.4 (en este caso $\lambda_i = 8$, marcado con una cruz roja). Partiendo de la raíz del árbol, se calcula iterativamente la distancia entre λ_i y los puntos de anclaje de cada subdominio, hasta llegar a alguna de las hojas del árbol. El camino seguido se marca en color rojo.

En la Figura 2.6 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.

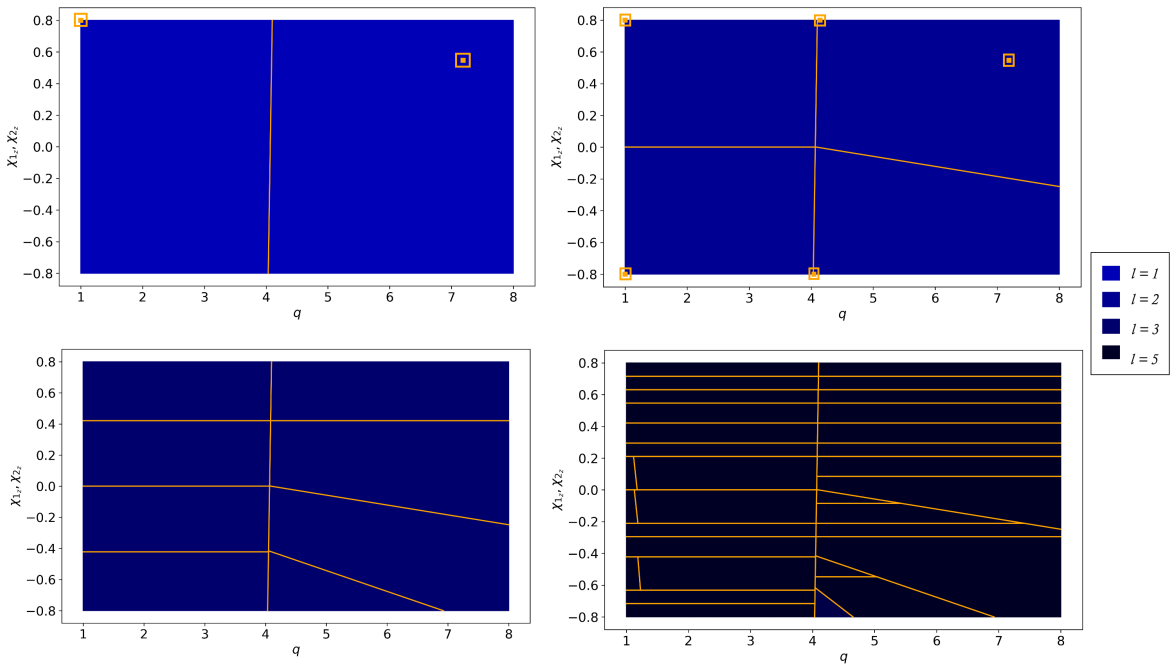

Figura 2.6: Ejemplo de partición del espacio de parámetros bidimensional para $l_{max} = 1, 2, 3$ y 5. En los primeros dos casos se muestran los puntos de anclaje en naranja.

En la Figura 2.7 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 velocidad de convergencia es claramente mayor en el segundo caso.

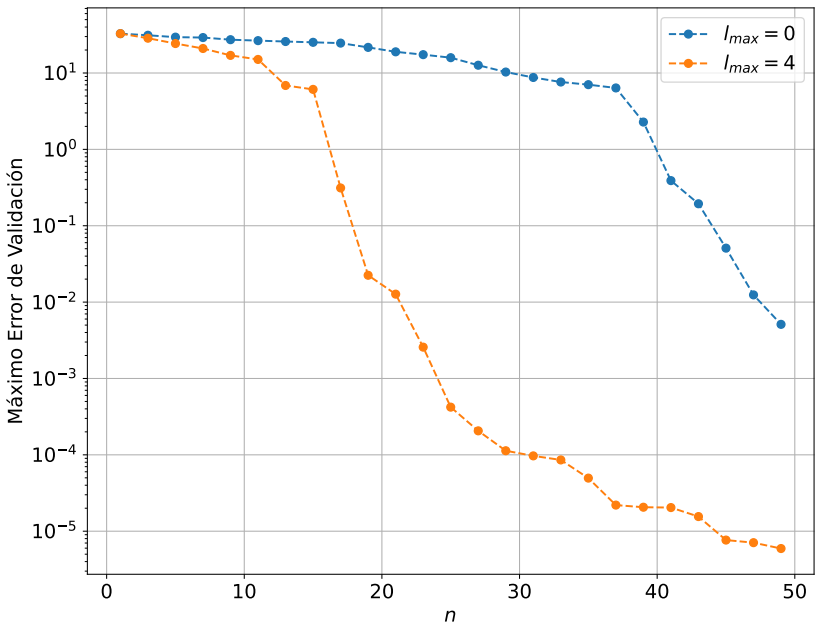

Figura 2.7: Base global ($l_{max} = 0$) versus base con $l_{max} = 4$ para distintos valores de n.

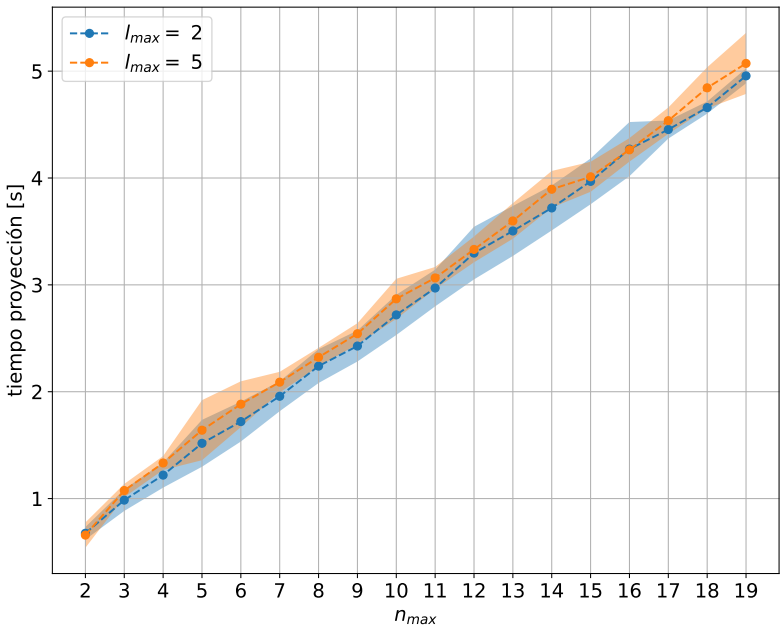

Figura 2.8: tiempos de proyección de un conjunto de validación a dos bases con distinto l_{max} en función del n_{max}. En cada caso la linea de trazo representa el valor medio, y el área de color indica una desviación estandar desde el valor medio, para cada medición.

En la figura 2.8 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}.

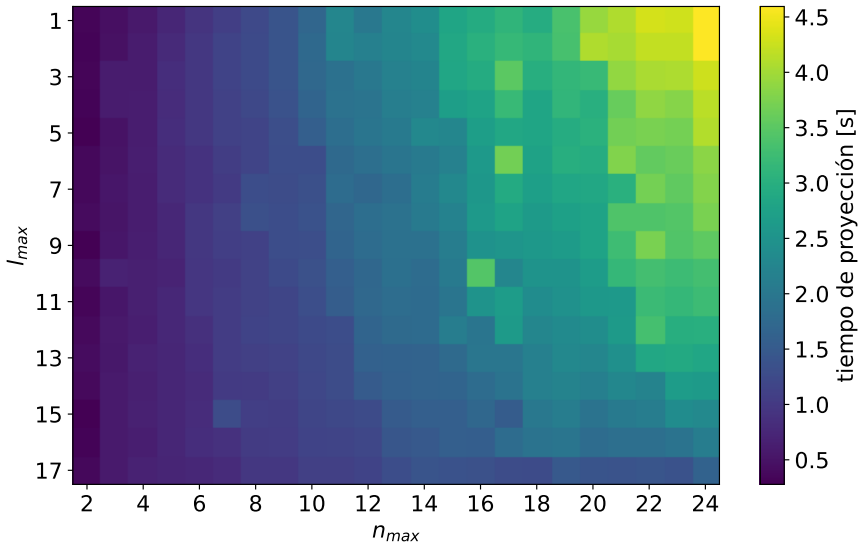

Figura 2.9: Tiempo de proyección de un conjunto de validación para diferentes valores de n_{max} y l_{max}

2.2.5. Tiempo de Proyección para distintos valores de n_{max} y l_{max}

Para representar una función de onda a partir de una base *hp-greedy* primero se debe buscar el subdominio (la hoja) correspondiente utilizando los puntos de anclaje y luego proyectar la función a la base local encontrada.

En este proceso, el trabajo de cómputo relevante es la proyección a la base local (que es independiente de l_{max}), que involucra calcular n coeficientes para una base local de n elementos. Por lo tanto el tiempo de proyección será afectado por n_{max} en mucha mayor medida que por l_{max}.

En la figura 2.9 se puede ver el tiempo de proyección para diferentes valores de n_{max} y l_{max}. En los primeros valores de l_{max}, con $l_{max} \leq 5$, 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.

Este comportamiento es consecuencia de la baja cantidad de elementos en el conjunto de entrenamiento en relación a la cantidad total de elementos en las bases hojas. Suponiendo un árbol denso (todas las bases tienen n_{max} elementos y todas las hojas tienen profundidad l_{max}), habrá un total de $n_{max} \times 2^{l_{max}}$ elementos entre todas las hojas.

Para obtener la Figura 2.9 se utilizó un conjunto de entrenamiento con 1400 funciones de onda, por lo que al llegar a unos valores de $l_{max} = 6$ y $n_{max} = 24$ en total

debería haber 1536 elementos entre todas las bases locales de las hojas. Es decir, más elementos de base que ondas en el conjunto de entrenamiento. Por lo tanto al aumentar el l_{max} llegará un punto en el que no habrá suficientes elementos en el conjunto de entrenamiento para que cada base local tenga n_{max} elementos. De forma que las bases locales serán cada vez más pequeñas, lo que dará lugar a un tiempo de proyección menor.

2.2.6. Hiperparámetros

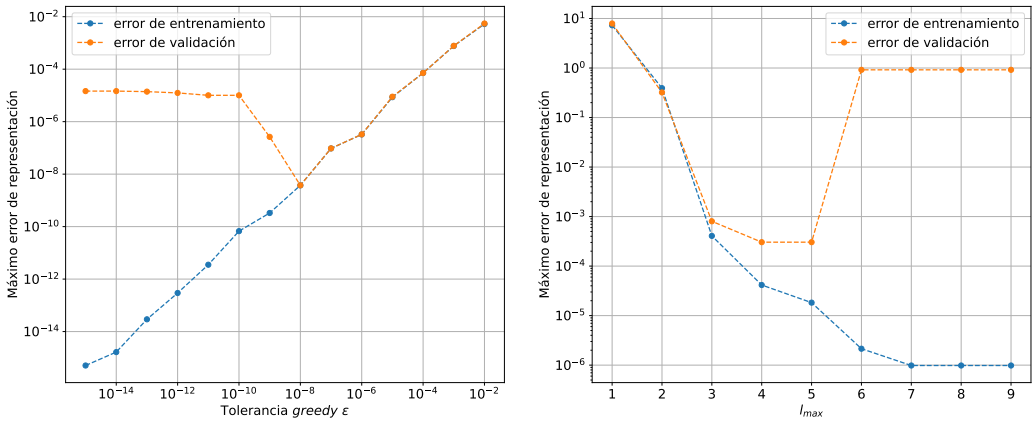

Figura 2.10: Ejemplos de *sobreajuste*. A la izquierda variando ε con $(n_{max}, l_{max}) = (25, 19)$ en un conjunto de parámetro unidimensional $(\lambda_i = q_i)$. A la derecha variando l_{max} con $(n_{max}, \varepsilon) = (20, 1 \times 10^{-6})$ en un conjunto de parámetro bidimensional$(\lambda_{ij} = (q_i, \chi_{z_j}))$.

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 junto a cierta combinación de hiperparámetros pueden dar lugar a sobreajustes en el modelo, un ejemplo de esto se puede ver en la figura 2.10. 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;

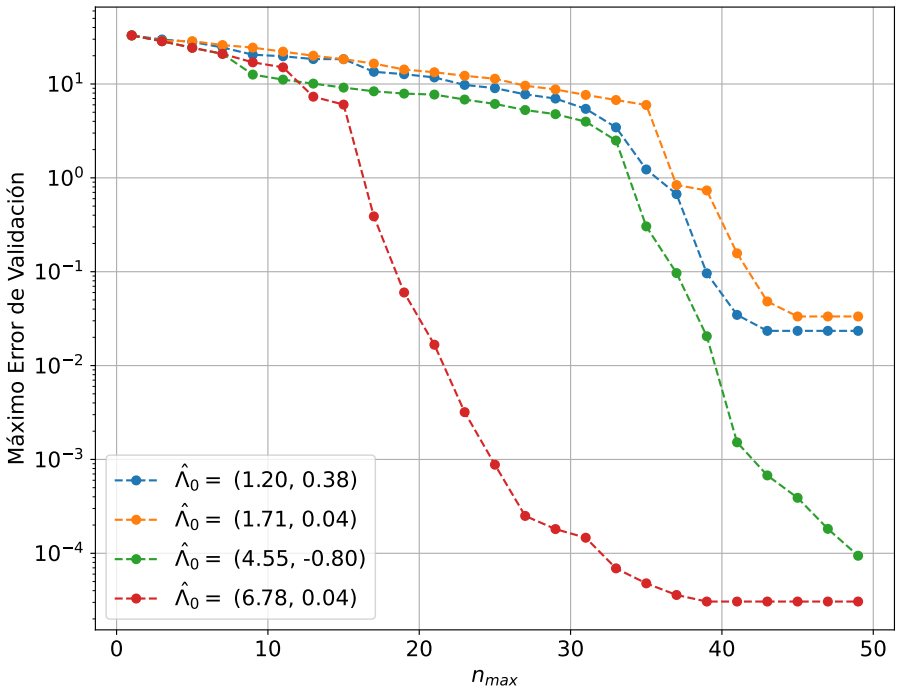

Figura 2.11: Error de validación para diferentes semillas $\hat{\Lambda}_0 = (q_0, \chi_{z_0})$.

- $\hat{\mathbf{\Lambda}}_\mathbf{0}$: la semilla 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 2.11 se puede ver como cuatro semillas diferentes dan lugar a cuatro curvas de error con distinta convergencia. En la figura 2.12, 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.

En caso de trabajar con un espacio de parámetros bidimensional la semilla será $\hat{\Lambda}_0 = (q_0, \chi_{z_0})$, con q la relación entre masas y χ_z el espín en la dirección z.

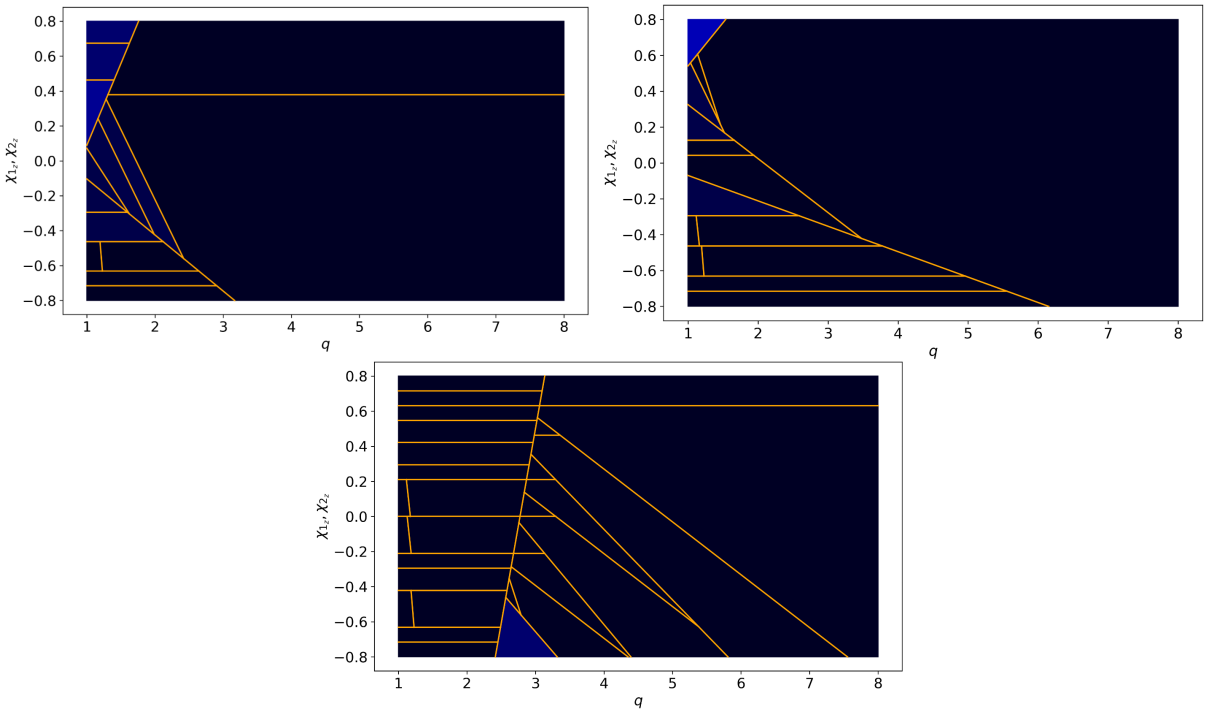

Figura 2.12: Partición del espacio de parámetros para tres semillas diferentes; a la izquierda $\hat{\Lambda}_0 = (1.2,\ 0.38)$, a la derecha $\hat{\Lambda}_0 = (1.71,\ 0.04)$ y al centro $\hat{\Lambda}_0 = (4.55, -0.8)$. Recordar que $\hat{\Lambda}_0 = (q_0, \chi_{z_0})$.

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** [13, 14], 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**[26, 27] (*Secuential Model-Based Optimization*).

Además existen dos métodos muy utilizados que no utilizan modelos, los cuales son la **búsqueda exhaustiva** (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

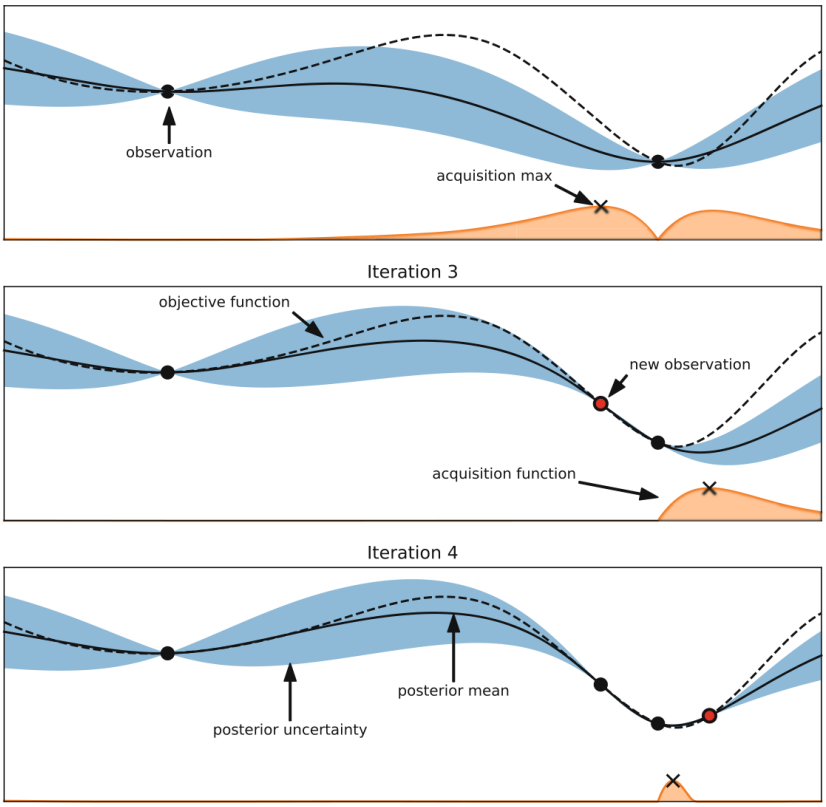

Figura 3.1: En la figura se observan tres iteraciones de una optimización bayesiana para una función sencilla con parámetro unidimensional. En linea punteada está representada la función real, mientras que con linea gruesa se representa el valor medio del modelo estadístico (en este caso construido utilizando 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 [28].

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.

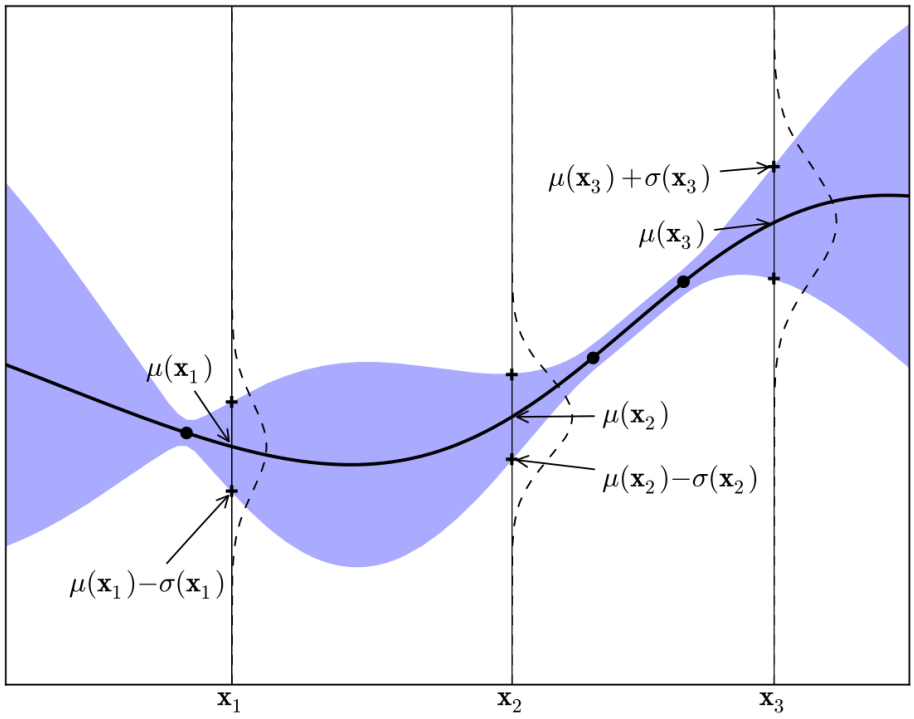

Figura 3.2: Proceso Gaussiano unidimensional con tres observaciones representadas por los puntos negros. La linea gruesa representa la media del modelo predictivo y la zona azul la varianza en cada caso. Se representa con linea de trazo las distribuciones normales para los valores x_1, x_2, y x_3[14].

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. En su lugar se utiliza el estimador de Parzen con estructura arbórea, que se explicará en la sección 3.2.3.

Para una introducción a la optimización bayesiana con procesos gaussianos ver [14].

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 *Expected Improvement* (EI) [29]. 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*) [27] 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 $P(y < y^*) = \gamma$.

En pocas palabras, TPE divide el esapcio de hiperparámetros en dos; uno "bueno", representado por $\ell(x_i)$, y otro "malo", representado por $g(x_i)$.

Mejora Esperada con TPE

Aplicando la ecuación (3.2) a la definición de mejora esperada (3.1) se obtiene la siguiente relación [27]:

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

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', x_p es un valor fijo *prior*, w_p es un peso *prior* y k es la función *kernel*, que en este caso son distribuciones gaussianas truncadas $\mathcal{N}_{trunc}(\mu, \sigma, a, b)$ centradas en los puntos x_i' (es decir, $\mu = x_i$) con a y b límites inferior

y superior del dominio. Por defecto los pesos $w_{x_i'}$ y w_p son iguales a 1. Para ver más detalles ver [30] y [27].

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 requiere 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 muestreo 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*** [15] 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.

Paralelización

El algoritmo escrito en Optuna permite paralelizar el proceso de optimización de forma asincrónica; esto se logra utilizando un método llamado *mentiroso constante* (*constant liar*) [27], el cual consiste en que al proponer un valor $\mathbf{x}^*$ se asigne temporalmente una evaluación falsa igual al promedio de los valores de y medidos, que luego se actualizará al valor obtenido al momento de finalizar la evaluación real. Este proceso puede hacer la optimización menos eficiente con respecto al número de iteraciones, pero más rápida a fin de cuentas ya que se pueden realizar varias iteraciones a la vez.

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, 31].

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 debe a que se espera que un menor tiempo de proyección de lugar a un modelo más rápido de evaluar, que es el objetivo planteado a largo plazo.

Esta optimización 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 objetivo es más importante en relación 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

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

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 imágenes 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, que se ilustra en la figura 3.3.

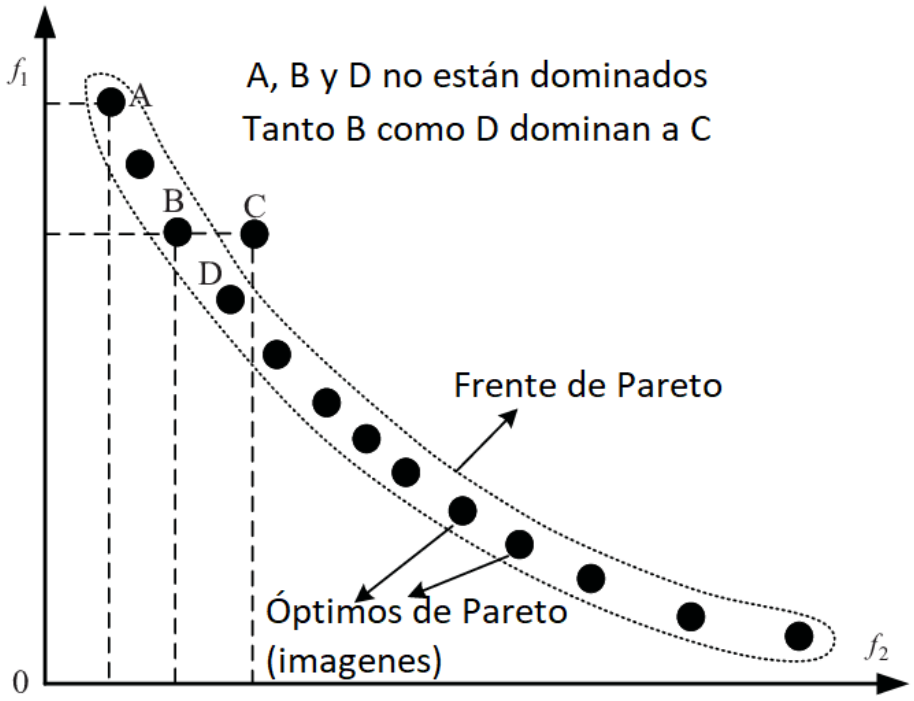

Figura 3.3: Ilustración del frente de Pareto [32].

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.

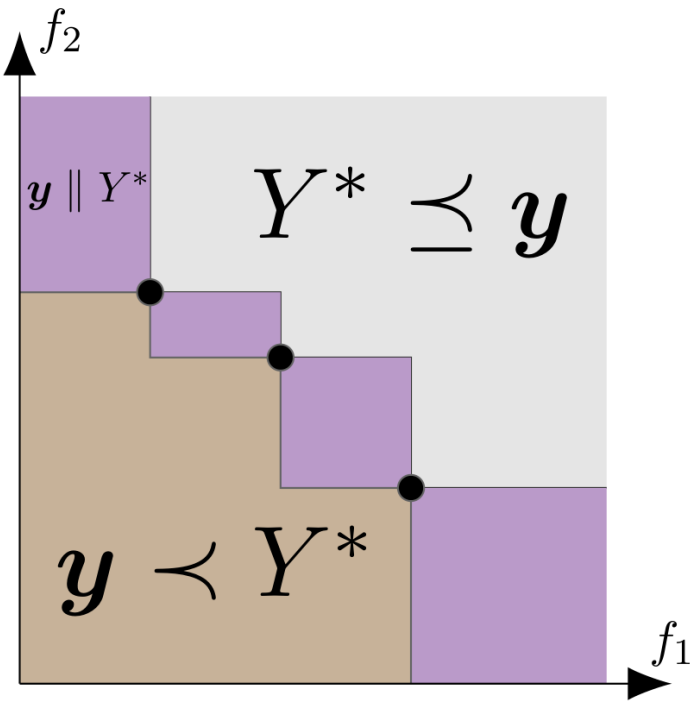

Figura 3.4: Relación entre $\mathbf{y}$ y el conjunto Y^*. Los puntos negros pertenecen a Y^*. La distribución $g(x_i)$ se construye con todos los puntos que caigan en la zona gris, y $\ell(x_i)$ con los puntos de las zonas restantes [30].

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 [30].

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 [15]). En linea oscura se representa la media del mejor error a cada iteración, y la zona sombreada representa

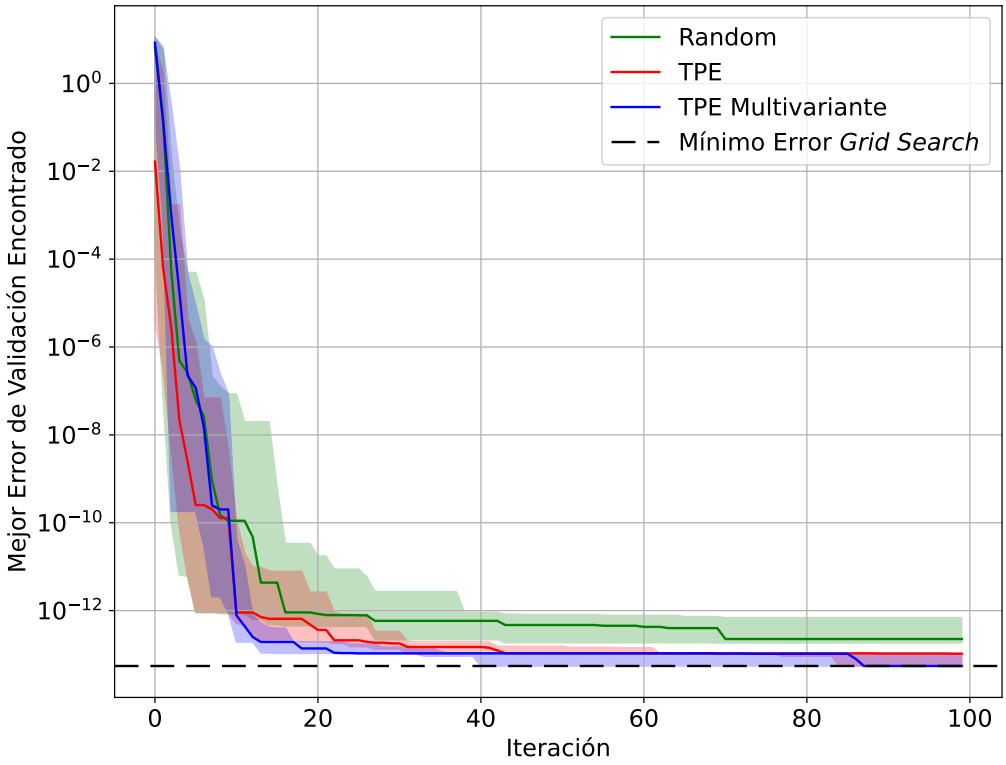

Figura 4.1: Comparación de convergencia. Se muestran los cuartiles para 20 optimizaciones realizadas, en cada caso.

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 multivariante) 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 $N \times L = \{(n_{max}, l_{max}) | n_{max} \in N, l_{max} \in L\}$.

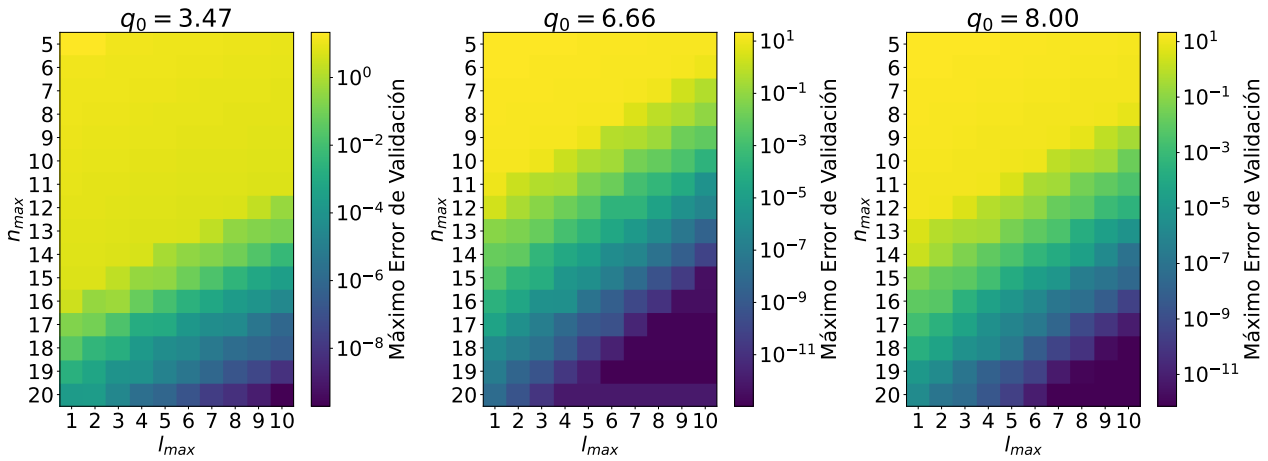

Figura 4.2: Máximo error de validación en función de n_{max} y l_{max} para tres diferentes semillas q_0.

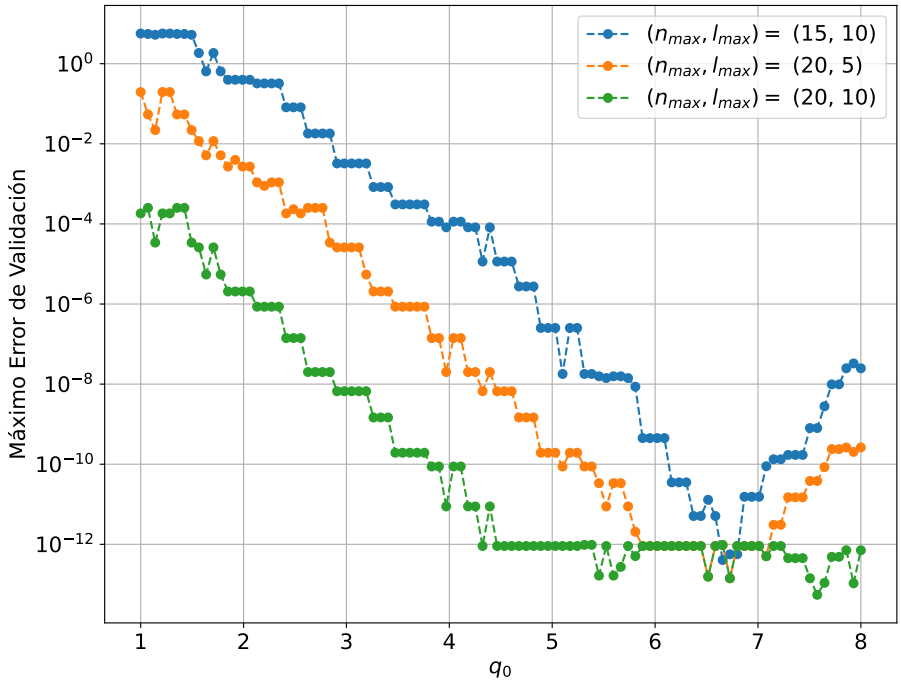

Figura 4.3: Máximo error de validación en función de la semilla q_0 para distintas combinaciones de (n_{max}, l_{max})

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.

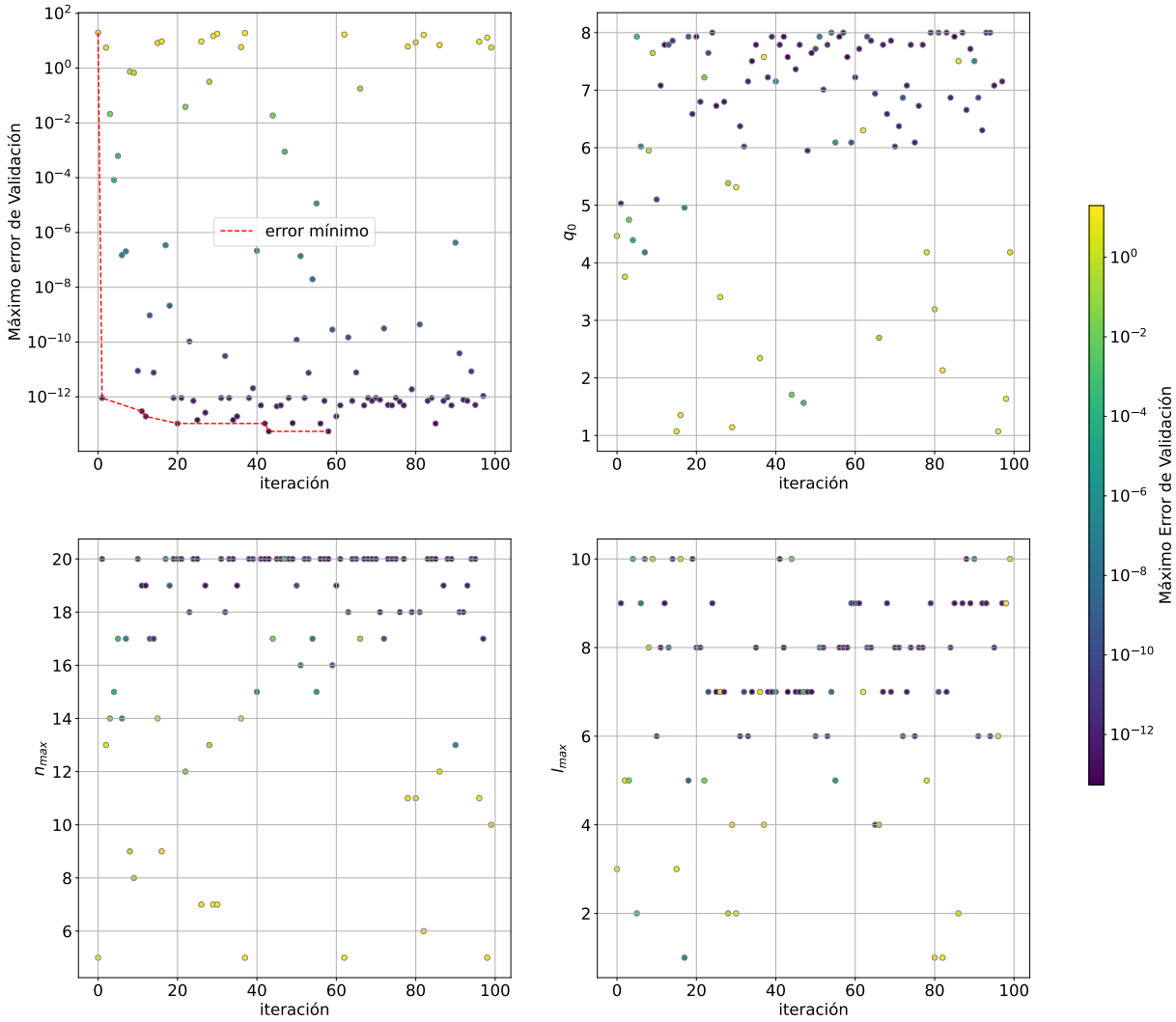

Figura 4.4: Optimización con 100 iteraciones utilizando el algoritmo TPE multivariante para el conjunto de entrenamiento con semilla unidimensional.

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 valores 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

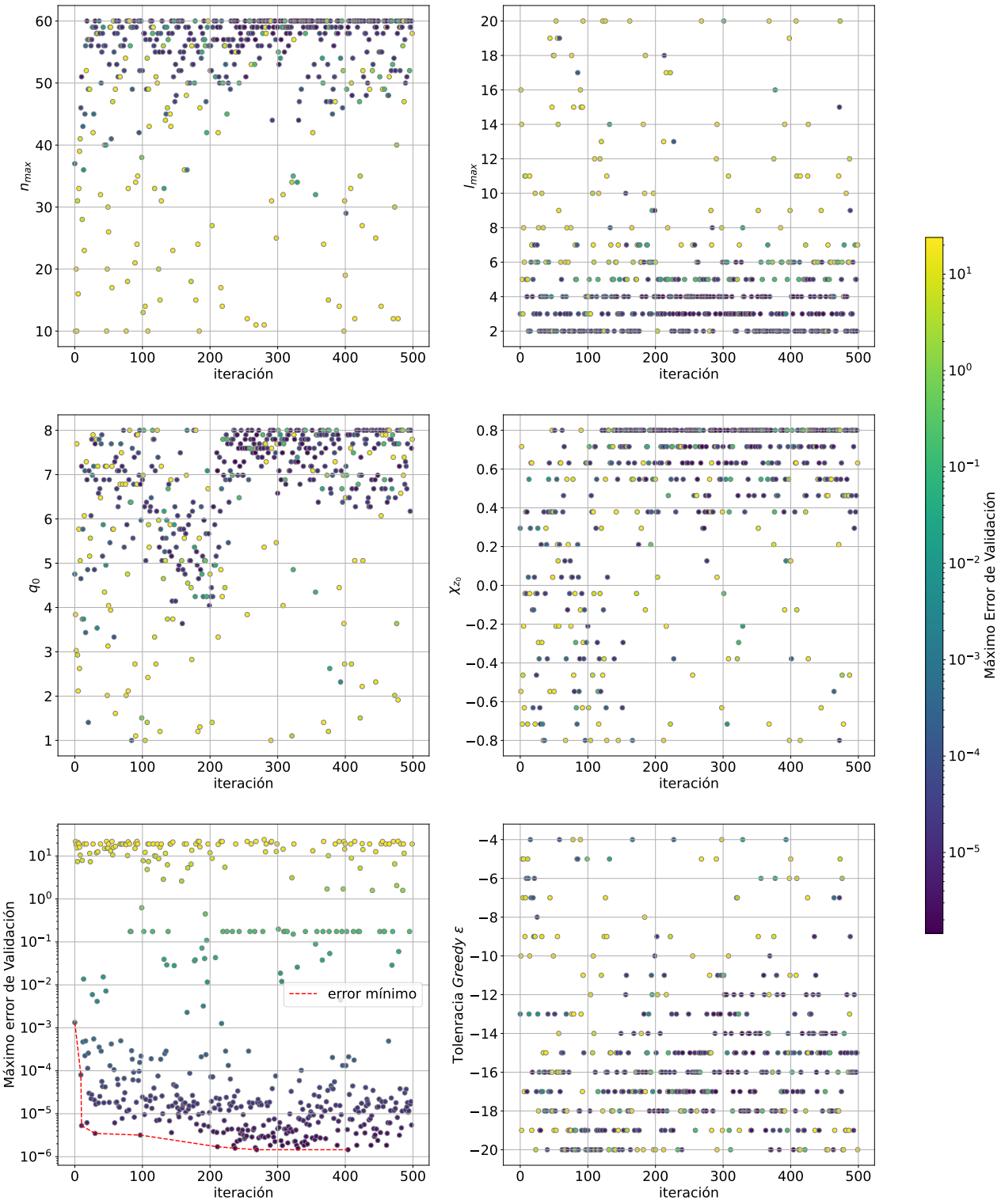

Figura 4.5: Optimización con 500 iteraciones para semilla de dos dimensiones utilizando el algoritmo TPE Multivariante con 4 trabajadores en paralelo.

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.

Error de Prueba

Luego de realizar la optimización se puede utilizar un conjunto de prueba, es decir, un conjunto de elementos que no se usaron para el entrenamiento ni para la optimización, para poner a prueba el resultado obtenido.

En este caso, utilizando un conjunto de prueba con 6000 elementos como resultado de un muestreo de 150 valores para q y 40 para χ_z, se calculó el máximo error relativo, siendo:

$$error\ relativo = \frac{\|h_\lambda - P_n h_\lambda\|^2}{\|h_\lambda\|^2}$$

Como resultado, se obtuvo un máximo error relativo de $3{,}84 \cdot 10^{-7}$

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, pero en general se observó que n_{max} y l_{max} suelen tener mayor importancia relativa al resto de hiperparámetros.

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

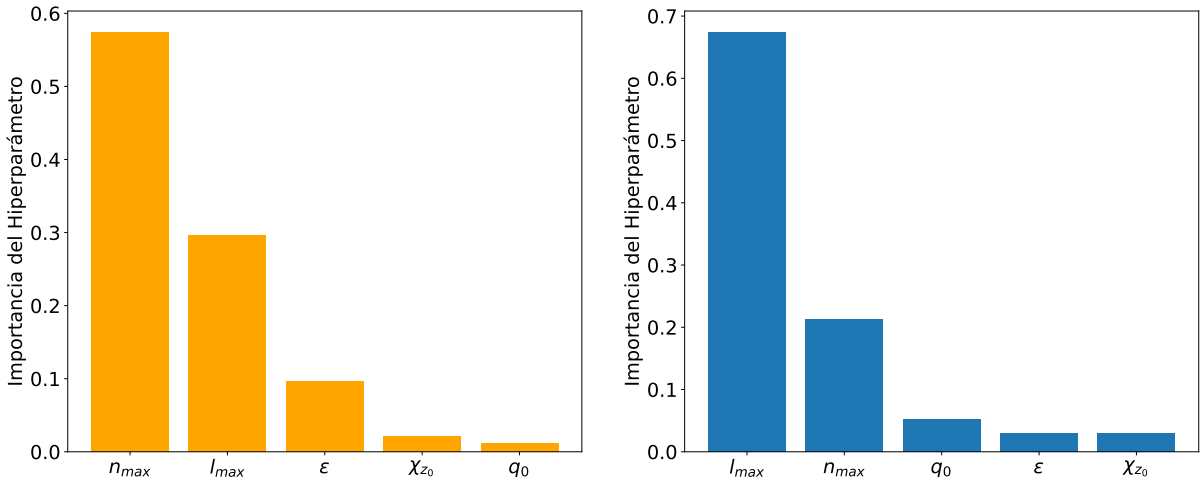

Figura 4.6: Importancia relativa de los hiperparámetros para dos espacios de búsqueda diferentes. n_{max} y l_{max} suelen ser los hiperparámetros más importantes en la mayoría de los espacios de búsqueda.

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.

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 y n_{max}. Mas bien, como ya se observó en la figura 2.8 de la sección 2.2.4 sobre 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 ($n_{max} \cdot 2^{l_{max}} < N$).

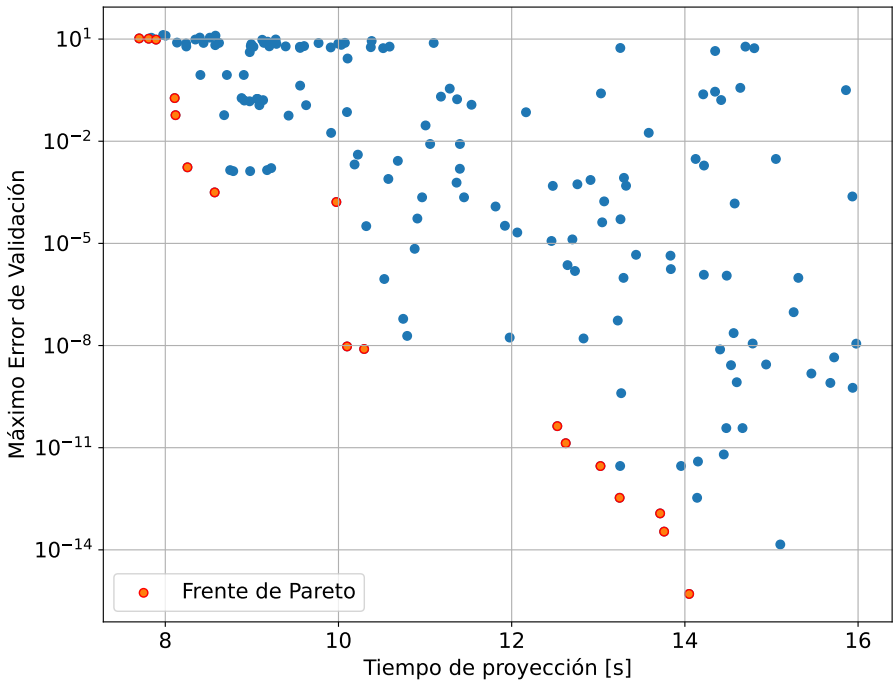

Figura 4.7: Máximo error de validación versus el tiempo de proyección. Cada punto representa una observación. En color naranja se marca el frente de Pareto.

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}.

Para entender la dependencia entre n_{max} y el tiempo de proyección hay que recordar la ecuación (2.3) de los coeficientes $c_{i,\lambda}$:

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

Para proyectar una función de onda h_λ a una base reducida se deben calcular los coeficientes $c_{i,\lambda}$, y el número de coeficientes será igual al número de elementos de la base $\{e_i\}$. Por lo tanto mientras más elementos tengan las bases locales, mayor será el tiempo de proyección.

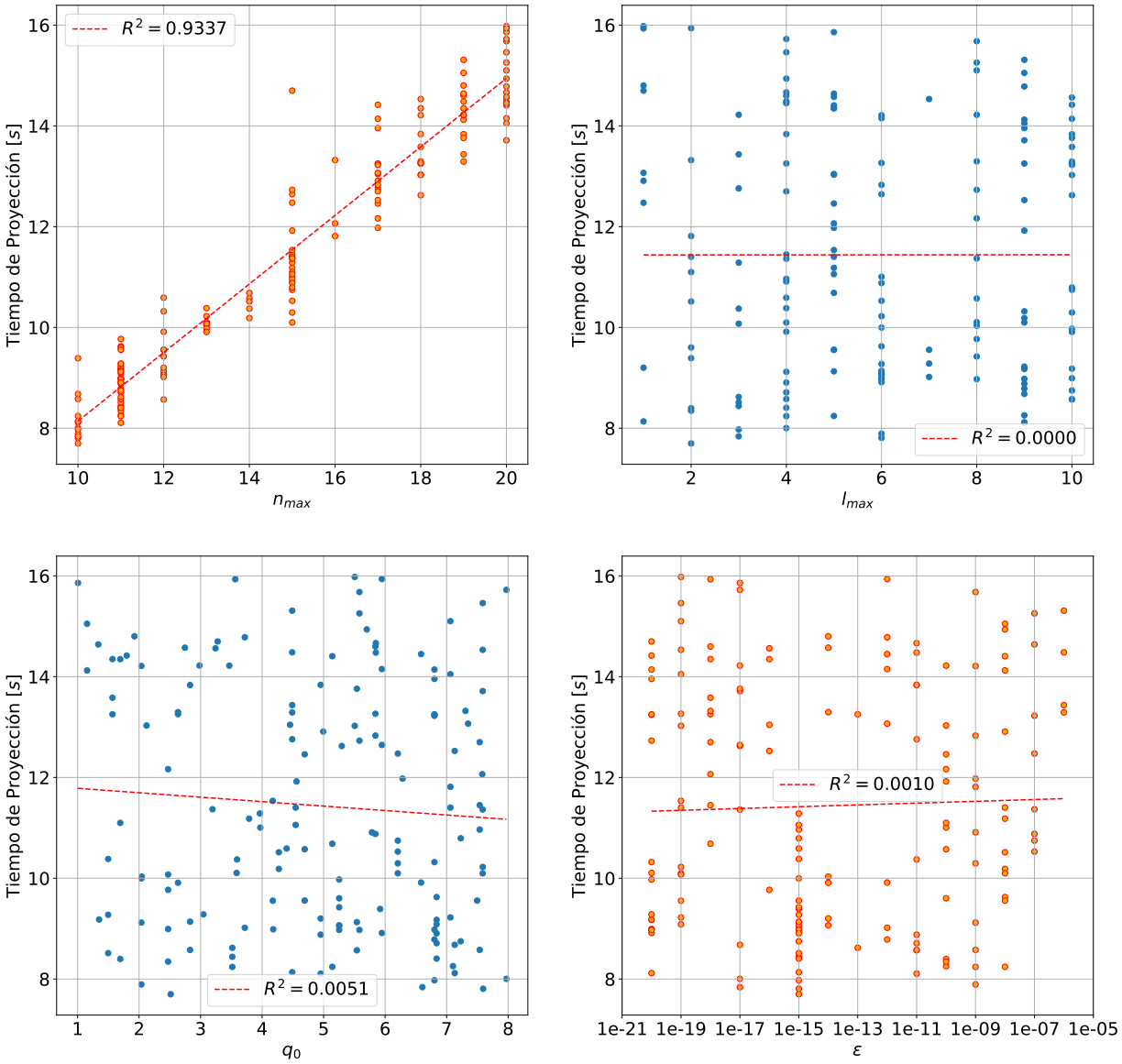

Figura 4.8: Relación entre el tiempo de ejecución y los diferentes hiperparámetros para una optimización de 160 iteraciones.

Capítulo 5

Conclusiones

El problema de las bases reducidas *hp-greedy*, que no está presente en la construcción de una base reducida global, es la complejidad añadida debido al gran número de configuraciones de hiperparámetros que afectan al rendimiento final de la base a la hora de representar un dado espacio.

Este problema se atacó utilizando métodos de optimización bayesiana, o más especificamente, el algoritmo TPE implementado en la librería Optuna. Los resultados de la optimización fueron muy alentadores, con una diferencia de varios ordenes de magnitud entre la primera configuración probada (elegida de forma aleatoria) y la mejor configuración encontrada por el algoritmo. Además se comparó este método con la optimización por búsqueda aleatoria, obteniendo mejores resultados con la optimización bayesiana.

Un aspecto negativo que salta a la luz es el incremento en el tiempo necesario para entrenar una base *hp-greedy* teniendo en cuenta el tiempo de optimización. Esto repercutirá directamente en el tiempo necesario para entrenar el modelo predictivo final. Es decir, que si bien se espera construir un modelo más rápido, el tiempo de entrenamiento de este aumentará notablemente.

Por último, la optimización multiobjetivo no resultó de mucho interés en este contexto debido a la gran correlación entre n_{max} y el tiempo de proyección, como se explica en la sección 4.2.2.

En conclusión, la optimización de hiperparámetros demostró ser un componente fundamental en la construcción de una base reducida *hp-greedy*, logrando muy buenos resultados aplicando optimización bayesiana, o más específicamente, el algoritmo TPE implementado en Optuna.

Trabajo a Futuro

Si bien las optimizaciones realizadas mostraron muy buenos resultados, aún se podría investigar la robustez de los hiperparámetros con métodos de análisis de sen-

sibilidad (los cuales son costosos de evaluar). Una configuración de hiperparámetros puede ser óptima pero inestable, es decir que se pierde la optimalidad al desplazarse un valor tan pequeño como se quiera. Esto no ocurre en el caso de configuraciones que sean óptimas y robustas. Por lo tanto un trabajo a futuro será analizar la robustez de los valores óptimos obtenidos con tests de análisis de sensibilidad.

Además, como se mencionó en la introducción, la construcción de bases reducidas constituye solamente el primer paso en la construcción de un modelo sustituto de orden reducido. Por lo tanto aún queda el trabajo de implementar el modelo predictivo final.

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. 1, 9, 12

[7] Hesthaven, J., Rozza, G., Stamm, B. Certified Reduced Basis Methods for Parametrized Partial Differential Equations. 2016. 1

[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. 1

[12] 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, 9, 14, 17

[13] 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, 25

[14] 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, 25, 28

[15] 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. 2, 30, 35

[16] 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, 4

[17] 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. 4

[18] Pinkus, A. n-widths in approximation theory. 1985. 10

[19] 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. 12

[20] 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. 12

[21] 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. 12

[22] 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. 12

[23] DeVore, R., Petrova, G., Wojtaszczyk, P. Greedy algorithms for reduced bases in banach spaces, 2012. URL https://arxiv.org/abs/1204.2290. 14

[24] 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. 14, 15

[25] Hesthaven, J. S., Gottlieb, S., Gottlieb, D. Spectral Methods for Time-Dependent Problems. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2007. 14

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

[27] 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. 25, 29, 30

[28] 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. 26

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

[30] 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. 30, 31, 33

[31] 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. 31

[32] Cai, Q., Lijia, M., Gong, M. A survey on network community detection based on evolutionary computation. *International Journal of Bio-Inspired Computation*, **8**, 01 2014. 32

[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. 36

[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. 41

Agradecimientos

Agradezco a mi familia, por darme las condiciones y el apoyo para hacer una carrera universitaria. A mis compañeros, que fueron mi ejemplo a seguir y de quienes nunca dejé de aprender cosas nuevas. A mi director Manuel Tiglio, y mi codirector Carlos Figueroa, por haberme dado la oportunidad de trabajar en esta tesis. Y a mis amigos, por siempre estar ahí.