Max-value Entropy Search for Multi-objective Bayesian Optimization with Unkown Constraints

Daniel Fernández Sánchez

Máster en Investigación e Innovación en Inteligencia Computacional y Sistemas Interactivos

MÁSTERES DE LA UAM

2019 - 2020

Escuela Politécnica Superior

UAM Universidad Autónoma de Madrid

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Max-value Entropy Search for Multi-objective Bayesian Optimization with Unkown Constraints

Máster Universitario en Investigación e Innovación en Inteligencia Computacional y Sistemas Interactivos

Author: Daniel Fernández Sánchez

Advisor: Daniel Hernández Lobato
Departamento de Ingeniería Informática

september 2020

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Optimización Bayesiana basada en la entropía del máximo valor para problemas con múltiples objetivos y restricciones desconocidas

Máster Universitario en Investigación e Innovación en Inteligencia Computacional y Sistemas Interactivos

Autor: Daniel Fernández Sánchez

Tutor: Daniel Hernández Lobato
Departamento de Ingeniería Informática

septiembre 2020

Daniel Fernández Sánchez
Optimización Bayesiana basada en la entropía del máximo valor para problemas con múltiples objetivos y restricciones desconocidas

Daniel Fernández Sánchez
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

En primer lugar, agradezco enormemente el uso de los medios del *Centro de Computación Científica* (*CCC*) de la Universidad Autónoma de Madrid, los cuales me han permitido realizar todos los experimentos y pruebas. Asimismo me gustaría agradecer la ayuda económica que me ha ofrecido la Universidad Autónoma de Madrid a través de la *Convocatoria de Ayudas para el fomento de la Investigación en Estudios de Máster-UAM 2019*, porque gracias a ella he podido dedicarme por completo a las asignaturas del Máster.

Por otro lado, doy las gracias a todos los profesores del máster que me han ayudado y guiado durante este año, y en especial a mi tutor, Daniel Hernández Lobato, por ayudarme y guiarme con todos los problemas que he tenido durante la elaboración de este trabajo incluso cuando estaba de vacaciones.

Finalmente, quiero agradecer mi familia por todo su apoyo, a mis amigos por no desesperarse aunque evitase todos los planes que me proponían (todavía tenemos que echarnos un Catán), y a mis compañeros de prácticas de este Máster por sus esfuerzos.

Resumen

Existe una gran cantidad de problemas donde el objetivo es mejorar varias cualidades de un sistema o producto. Sin embargo, no se dispone de la expresión analítica de la función objetivo que define los valores de esas cualidades, lo que dificulta su optimización ya que es difícil calcular su gradiente. Además, muchas veces evaluar nuevas configuraciones de parámetros iniciales es una tarea muy costosa que requiere de bastante tiempo. Asimismo, los valores obtenidos de cada configuración, en las funciones objetivo, suelen estar contaminados por ruido, dado que no es posible medir con suficiente precisión o hay defectos en la fabricación. Por otro lado, también es posible que se deban cumplir ciertas restricciones. Por ejemplo, se podría intentar maximizar la velocidad de un robot al mismo tiempo que se quiere minimizar su consumo y no romper ninguna de sus articulaciones. Se puede ver que la velocidad y el consumo energético son dos objetivos conflictivos, dado que mejorar en uno implica empeorar en el otro. También es posible percatarse de que el peso y la composición de los materiales del robot influirán en el cumplimiento de las restricciones, dado que si se rompe una articulación los valores obtenidos por esa configuración serán inválidos. Por otro lado, se debe observar que construir nuevos robots es una tarea muy costosa, así que lo ideal sería reducir su cantidad. Se puede apreciar que, cuando hay varios objetivos implicados no suele existir una única solución al problema, sino que hay un conjunto de soluciones que son las mejores en el balance que consiguen de los objetivos. En el ejemplo del robot, una solución sería el robot más rápido pero que más consume y otro el más lento pero que menos energía necesita. Para resolver este tipo de problemas con múltiples objetivos sujetos a varias restricciones desconocidas, donde no se dispone de la expresión analítica para ninguno de ellos, se puede utilizar optimización Bayesiana.

La optimización Bayesiana cuenta con dos piezas clave para resolver estos problemas. Por un lado un modelo probabilístico de las funciones objetivo y restricciones, y por otro una función de adquisición, que se encarga de evaluar la utilidad de realizar nuevas evaluaciones. En este trabajo se desarrolla la función de adquisición MESMOC, capaz de afrontar estos problemas multi-objetivo con varias restricciones. Esta función de adquisición reduce la entropía del óptimo valor asociado a la solución del problema a fin de obtener con el menor número de evaluaciones una potencial solución al problema.

Con la finalidad de ver el desempeño de MESMOC, se han realizado dos experimentos tanto con datos sintéticos como con reales, en los que se ha comparado el método desarrollado con otros del estado del arte como PESMOC. En estos experimentos se ha visto que MESMOC es competitivo con estos métodos, pero es más rápida de evaluar.

Palabras clave

Optimización Bayesiana, Múltiples objetivos y restricciones, Procesos Gaussianos

Abstract

There are a lot of problems where the goal is to improve several qualities of a system or product. However, the analytical expression of the objective function that defines the values of those qualities is not available, making it difficult to optimize them since it is difficult to calculate their gradient. In addition, many times evaluating new initial parameter settings is a very expensive and time-consuming task. Furthermore, the values obtained from each configuration, in the objective functions, are usually contaminated by noise, since it is not possible to measure with sufficient precision or there are defects in manufacturing. On the other hand, certain restrictions may also have to be met. For example, we could try to maximize the speed of a robot at the same time that we want to minimize its consumption and we would like to avoid breaking any joints. We can see that the speed and the energy consumption are two conflicting objectives because to improve in one implies to worsen in the other. Also, it is possible to realize of that the weight and the composition of the materials of the robot will influence in the fulfilment of the restrictions because if we break one articulation the values of the objectives of that configuration will be invalid. On the other hand, we can see that building new robots is a very expensive task, so the ideal would be to reduce their number. We can also see that, when there are several objectives involved, there is usually not a single solution to the problem, but there is a set of solutions that attain the best trade-off between the two objectives. In the example of the robot, one solution would be the fastest but will lead to more energy consumption and other the slowest but less consuming robot. To solve this type of problems with multiple objectives subject to several unknown constraints, where the analytical expression is not available for any of them, we can use Bayesian optimization.

Bayesian optimization has two key pieces to solve these problems. On one hand a probabilistic model of the objective functions and constraints, and on the other hand an acquisition function, which evaluates the utility of performing new evaluations. In this work, the acquisition function MESMOC is developed, capable of facing these multi-objective problems with several restrictions. This acquisition function reduces the entropy of the optimal value associated with the solution of the problem in order to obtain with the minimum number of evaluations a potential solution to the problem.

In order to see the performance of MESMOC, two experiments have been carried out with both synthetic and real data, in which the method developed has been compared with others of the state of the art as PESMOC. In these experiments, it has been seen that MESMOC is competitive with these methods, but it is faster to evaluate.

Keywords

Bayesian optimization, Constrained Multiobjective problems, Gaussian processes

ÍNDICE

Listas

Lista de algoritmos

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación

En la actualidad existen multitud de problemas de optimización. En ellos, el objetivo es variar los parámetros iniciales con el propósito de mejorar algún sistema o producto. Por ejemplo, si se quisiera maximizar la velocidad de una turbina, se podría variar las proporciones de los materiales que la componen, ajustar los tamaños de los engranajes o modificar los compuestos de su fuente de alimentación. Sin embargo, encontrar la mejor configuración para todos los parámetros es complicado. Además, lo ideal sería minimizar la cantidad de prototipos a probar, dado que construir nuevos prototipos de turbina es una tarea muy costosa y que requiere bastante tiempo.

Bayesian optimization (BO) es una técnica que ha demostrado muy buenos resultados en problemas como el descrito anteriormente [1,2]. Este tipo de problemas se caracterizan por no disponer de la expresión analítica de la función objetivo (en este caso la velocidad de la turbina), así que, calcular su gradiente es difícil. Además, las evaluaciones podrían ser costosas tanto temporal como económicamente de realizar. Asimismo, estas evaluaciones podrían estar contaminadas por ruido, en el ejemplo de la turbina es posible que hubiera pequeños defectos en el prototipo. Para abordar estos problemas, BO realiza una búsqueda en el espacio de parámetros de forma inteligente, dado que utiliza las observaciones realizadas anteriormente para elegir cuál es la siguiente mejor configuración de parámetros a probar, es decir, el siguiente mejor punto del espacio de parámetros a evaluar. Para ello, BO cuenta con dos piezas clave, y supone que la función analítica de la caja negra (la función objetivo) es suave. La primera de las piezas clave de BO es un modelo estadístico que describe la función objetivo. La segunda es la función de adquisición, la cual se encarga de valorar la utilidad de evaluar en cada punto del espacio de parámetros, de forma que al maximizarla, se obtiene la siguiente mejor configuración de parámetros a probar. Típicamente se utilizan los procesos Gaussianos como modelo probabilístico de la función objetivo, los cuales serán tratados en el Capítulo 2. Así como en la Sección 3.2 se hablará de la función de adquisición. Por último, se debe tener en cuenta que, como el coste del evaluar en la caja negra es enorme, se considera que en comparación el coste computacional de ejecutar el algoritmo de BO es despreciable.

Por otro lado, existen otros problemas más difíciles que el descrito anteriormente. Dado que puede haber varios aspectos a mejorar al mismo tiempo, además de algunas restricciones que prohibirían los valores que se obtendrían de ciertas configuraciones de parámetros. Como hay varias funciones objetivo ya no hay un único punto en el espacio de entradas que sea óptimo, sino un conjunto potencialmente infinito de puntos óptimos. Este conjunto es conocido como conjunto de Pareto solución $\mathcal{X}^\star$ [3]. Un ejemplo de este tipo de problemas consistiría en diseñar una red neuronal para un problema de clasificación. En este problema, se podría modificar la cantidad de capas ocultas que tiene la red, el número de neuronas en cada capa, la regularización, el ratio de aprendizaje por cada iteración u otros parámetros. Asimismo, las funciones objetivo serían: minimizar el error en las predicciones de la red, maximizar la velocidad de sus predicciones y minimizar su consumo energético. Además, se deberían cumplir algunas restricciones que impedirían ciertas configuraciones de parámetros, aunque estas configuraciones son desconocidas a priori. Estas restricciones serían: la cantidad de memoria que utiliza la red y el tiempo que requiere su entrenamiento. Para afrontar este tipo de problemas, también se puede utilizar BO [4, 5], la cual, de forma equivalente a los problemas con una sola función objetivo, supone que todas las funciones objetivo y restricciones son cajas negras cuya expresión es suave.

La elaboración de una función de adquisición para problemas con múltiples objetivos y restricciones parte de una función de adquisición para un solo objetivo, a la que hay que aplicar algunas modificaciones. En la literatura existen múltiples funciones de adquisición para un solo objetivo [6–8]. Por un lado, estas han sido modificadas para abordar el problema de optimizar múltiples objetivos [9–11]. Al mismo tiempo, también han sido extendidas para trabajar con un objetivo y varias restricciones [12–14]. Estas dos adaptaciones son empleadas para formar las funciones de adquisición de problemas con múltiples objetivos y restricciones. Sin embargo, esta adaptación no es directa dado que ambas modificaciones deben ser compatibles entre si. BMOO es una función de adquisición capaz de hacer frente a estos problemas multi-objetivo con restricciones, pero tiene algunos inconvenientes que se comentan en la Sección 4.2.3. Una alternativa a la BMOO es la función de adquisición PESMOC. Esta función de adquisición también se puede usar en problemas multi-objetivos con restricciones pero no cuenta con los impedimentos de BMOO y además es más rápida [5]. PESMOC pertenece al grupo de funciones de adquisición, basadas en la teoría de la información, que tienen como objetivo minimizar la entropía (maximizar la información) de la solución del problema. Esta función de adquisición, se basa en la búsqueda de entropía predictiva y surge como una extensión de PESMO [15] y PESC [16], que a su vez son las adaptaciones de PES [17] para problemas con múltiples objetivos sin restricciones y problemas con un objetivo y varias restricciones, respectivamente.

Por otro lado, MES, al igual que PES, es una función de adquisición que reduce la entropía de la solución [18], pero realiza una búsqueda utilizando el máximo valor, lo que permite simplificar las cálculos, facilitar su implementación y acelerar el cálculo de la función de adquisición. Esta función de adquisición ya ha sido adaptada a problemas con múltiples objetivos (MESMO [19]), y a problemas con

un objetivo y restricciones (MESC [20]). Sin embargo, estas dos funciones de adquisición no pueden trabajar simultáneamente con varios objetivos y restricciones. En este trabajo se desarrolla MESMOC, una función de adquisición que puede afrontar problemas con múltiples objetivos y varias restricciones. En los experimentos del Capítulo 5 se puede ver que obtiene un rendimiento similar o un poco inferior a PESMOC, pero es mucho más rápida. Asimismo, al igual que este último, MESMOC también se puede utilizar de forma desacoplada.

1.2. Estructura del trabajo

A continuación, se describe la organización del resto del presente trabajo:

Procesos Gaussianos En el segundo capítulo se explicará la base teórica que tiene y cómo funciona el modelo estadístico que se utiliza en la optimización Bayesiana para modelar funciones objetivo o restricciones.

Optimización Bayesiana En el tercer capítulo, se explicará como utiliza la optimización Bayesiana el modelo de la función objetivo para reducir el número de evaluaciones a realizar mediante la función de adquisición.

Optimización Bayesiana con múltiples objetivos y restricciones desconocidas En el cuarto capítulo, primeramente se explicará cómo son los problemas con múltiples objetivos y restricciones. Seguidamente, se tratará el uso de la función de adquisición en problemas con múltiples objetivos y restricciones. Además, se explicará la función de adquisición PESMOC y la función de adquisición desarrollada MESMOC.

Experimentos realizados En el quinto capítulo se describirán tanto los experimentos sintéticos como los reales realizados, y se discutirán los resultados obtenidos.

Conclusiones y trabajo futuro En el último capítulo se recogerán las principales conclusiones que se pueden extraer del trabajo realizado, y se describirán posibles vías de trabajo futuro.

2

Procesos Gaussianos

Puesto que el objetivo de la optimización Bayesiana es utilizar las evaluaciones pasadas para elegir dónde realizar las futuras, se debe utilizar un modelo que prediga los posibles valores que habrá en regiones del espacio desconocidas, para así saber si merece la pena evaluar en esos puntos de la función objetivo. Los problemas donde la finalidad es predecir nuevos valores dados los anteriores se conocen como problemas de regresión. En ellos el objetivo es predecir el valor de un nuevo y_n, dados los anteriores valores $(X, \mathbf{y})$, y suponiendo que:

$$y_n = f(\mathbf{x}_n) + \epsilon_n, \qquad \epsilon_n \sim \mathcal{N}(0, \sigma_n^2)$$

donde $\mathbf{x}_n$ es un vector de entrada, $f(\mathbf{x}_n)$ es el valor de la función objetivo a predecir, ϵ_n es el ruido (es decir, una variable aleatoria generada de forma independiente de una distribución Gaussiana unidimensional con media 0 y varianza σ_n^2), e y_n es el valor observado dado el punto la entrada $\mathbf{x}_n$. Una primera aproximación para resolver este problema consistiría en aplicar un modelo de regresión lineal a los valores ya obtenidos, para así estimar cuál podría ser el valor de un punto en una región del espacio sin explorar. Sin embargo, esta no suele ser la mejor opción, dado que se desconoce la cantidad de parámetros que debería tener este regresor [21, Capítulo 1]. Además, este enfoque no permite conocer cuál es la incertidumbre de las evaluaciones en nuevos puntos, algo que es de gran importancia para guiar la búsqueda mediante BO, como se verá en los Capítulos 3 y 4. Conceptualmente es fácil ver que, si un nuevo punto de entrada es próximo a otro ya observado, su valor de salida será más similar que el de otro que sea más distante. Por tanto, se tiene menos incertidumbre sobre los valores que puede tomar.

Estos problemas se pueden solucionar si se utiliza como modelo de la función objetivo un proceso Gaussiano. Este modelo es no paramétrico, es decir, su número de parámetros crece según aumenta la cantidad de puntos observados $(\mathbf{x}_n, y_n)$, por lo que no hace falta preocuparse por su cantidad. Además, también da una medida sobre la incertidumbre, es decir, el desconocimiento que se tiene sobre los valores que pueden tomar diversos puntos en regiones no exploradas. Asimismo, la expresión que se deriva de ellos tiene una solución analítica, por lo que es rápida de evaluar. Por otro lado, los procesos Gaussianos se pueden aplicar a problemas tanto de clasificación como de regresión, pero en este trabajo solamente se han usado en este segundo ámbito.

2.1. Procesos Gaussianos en regresión

Los procesos Gaussianos, Gaussian processes (GPs), son una generalización de la distribución Gaussiana unidimensional en el ámbito de infinitas dimensiones. Su definición formal es la siguiente:

Definición 2.1 *Un proceso Gaussiano es una distribución de probabilidad sobre funciones, de tal manera que al escoger un conjunto finito de puntos y evaluar la distribución de probabilidad de la función en esos puntos, la distribución conjunta es una Gaussiana multivariante.*

Por lo tanto, en lugar de estar definidos por una media μ y una varianza σ^2, estos se definen mediante una función de medias $m(\mathbf{x})$ y una función de covarianzas $K(\mathbf{x}, \mathbf{x}')$:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2.1a}$$

$$K(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{2.1b}$$

y se escriben de la siguiente forma:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')). \tag{2.2}$$

Esta ecuación significa que el valor de la función objetivo $f(\mathbf{x})$ ha sido generado mediante un $\mathcal{GP}$, es decir, si generamos valores aleatorios del $\mathcal{GP}$ en $\mathbf{x}$, uno de ellos será el que toma la función objetivo en $\mathbf{x}$.

Como los procesos Gaussianos son una colección de infinitas variables aleatorias, esto implica que hay un requerimiento de consistencia entre ellas, también conocido como la propiedad de marginalización [22, Capítulo 2]. Esta propiedad quiere decir que si se tiene el vector (y_1, y_2), donde $(y_1, y_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\mu}$ es el vector de medias de y_1 e y_2, y $\boldsymbol{\Sigma}$ es su matriz de covarianzas. Entonces, se debe cumplir que $y_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$. Es decir, el comportamiento de un conjunto de variables aleatorias no modifica la naturaleza de un subconjunto de ellas. Tal como se dijo en la definición, al generar valores mediante cualquier subconjunto de variables aleatorias, se obtendrá una distribución de probabilidad conjunta que será una Gaussiana multivariante.

2.1.1. Distribución predictiva

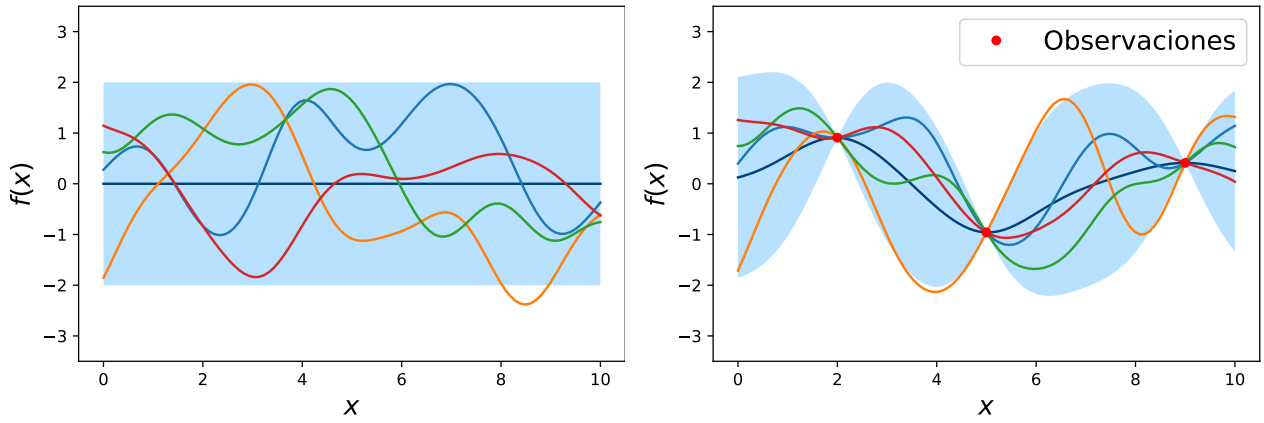
En los problemas de regresión se cuenta con un conjunto de datos $\mathcal{D} = \{(\mathbf{x}_n, y_n) \mid n = 1, ..., N\} = (X, \mathbf{y})$, se supone (2.1), y el objetivo es predecir los valores un vector $\mathbf{f}_*$, dada una nueva matriz de puntos de entrada X_*, donde $\mathbf{f}_* = f(X_*)$. Se quiere predecir $\mathbf{f}_*$ en lugar de $\mathbf{y}_*$ debido a que $\mathbf{y}_*$ está contaminado por ruido. De esta formulación surge naturalmente la necesidad de aplicar el teorema de Bayes, dado que se tiene una creencia a priori de los valores de los datos (el priori), unos datos

observados (la verosimilitud), y se quiere condicionar este priori a los datos con los que ya se cuenta con el propósito de predecir nuevos valores.

El priori será no informativo, para evitar sesgar las predicciones, por lo que su función de medias será $\mathbf{0}$ y su función de covarianzas $K(\cdot,\cdot)$:

$$p(\mathbf{f}|X) \sim \mathcal{GP}(\mathbf{0}, K(X,X)) \tag{2.3}$$

donde $K(X,X)$ es la matriz de covarianzas de X. En la Figura 2.1(a), se muestran cuatro funciones generadas aleatoriamente utilizando esta distribución a priori, y suponiendo que no hay ruido, esta suposición se ha mantenido para el resto de imágenes de este capítulo. Se puede ver que todas las funciones generadas son suaves, debido a la función kernel utilizada, en este caso squared exponential (SE), que es infinitamente derivable. En consecuencia, la forma de las funciones generadas con ella es suave. En la Sección 2.2 se hablará más en detalle de los kernels.



(a) Priori (b) Posteriori

Figura 2.1: En cada gáfica se muestran cuatro funciones generadas a partir de la distribución a priori 2.1(a) y la distribución a posteriori 2.1(b) (con tres observaciones y suponiendo que no tienen ruido). En ambos casos se utilizó el kernel squared exponential (SE), con $\ell = 1{,}0$ y $\sigma_f^2 = 1{,}0$. El área sombreada de color azul claro corresponde al intervalo de confianza del 95 % del $\mathcal{GP}$, y la línea azul oscura es la media del $\mathcal{GP}$. En 2.1(b) las observaciones que ya se tienen, a las que se ha condicionado el priori, se identifican con un punto rojo.

A medida que se van obteniendo datos, se condiciona el priori a estos para obtener el posteriori, y este será el priori de la siguiente iteración. La distribución conjunta del priori y la verosimilitud, cuando ya se conocen los valores $\mathbf{f}$ (suponiendo que se observan los verdaderos valores sin ruido) dadas las entradas X, y se desconocen los valores $\mathbf{f}_*$ dada la matriz X_*, es la siguiente:

$$p(\mathbf{f}, \mathbf{f}_*|X, X_*) = \begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} |X, X_* \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{2.4}$$

Para condicionar el priori a la verosimilitud, se pueden utilizar las ecuaciones que se derivan en [21,

Sección 2.3], obteniendo el siguiente resultado:

$$p(\mathbf{f}_*|X_*, X, \mathbf{f}) \sim \mathcal{N}(K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)). \tag{2.5}$$

Esta es la distribución a posteriori de los valores desconocidos $\mathbf{f}_*$ dadas las entradas X_*, y dadas las salidas ya conocidas $\mathbf{f}$ de las entradas X. Como esta distribución se utiliza para predecir valores desconocidos, se suele denominar distribución predictiva. A fin de simplificar la notación, se realizarán los siguientes cambios: $K_*^{\mathrm{T}} = K(X_*X)$, $K^{-1} = K(X, X)^{-1}$, $K_* = K(X, X_*)$ y $K_{**} = K(X_*, X_*)$. Tal como se dijo anteriormente, en la mayoría de los casos las observaciones están contaminadas por ruido, y en consecuencia, no se observan los valores $\mathbf{f}$ sino los $\mathbf{y}$. Para tener esto en cuenta, simplemente se debe sumar la varianza del ruido a los valores de la diagonal de la matriz K. Con estos cambios, la función de medias y la función de covarianzas de la distribución predictiva quedan de la siguiente forma:

$$m(X_*) = K_*^{\mathrm{T}}(K + \sigma_n^2 I)^{-1}\mathbf{y} \tag{2.6a}$$

$$v(X_*) = K_{**} - K_*^{\mathrm{T}}(K + \sigma_n^2 I)^{-1}K_*. \tag{2.6b}$$

Se puede ver que la función de covarianzas $v(X_*)$ no depende de los valores observados, sino solamente de las entradas X y X_*. Esto implica que el grado de certidumbre que arroja un punto sobre otro solamente está determinado por la distancia que hay entre ellos no sobre sus valores.

El coste computacional de los procesos Gaussianos viene dado por la inversión de la matriz de covarianzas $(K + \sigma_n^2 I)$, el cual es $\mathcal{O}(N^3)$ para una matriz de $N \times N$. Se pueden utilizar métodos más rápidos que la inversión estándar, como invertir usando la descomposición de Cholesky, que es más estable, pero el coste continúa siendo del mismo orden.

2.2. Kernels

La función de covarianzas de los procesos Gaussianos es un kernel. Los kernels también son utilizados por otros modelos de aprendizaje automático, como las máquinas de vectores de soporte, support vector machines (SVMs). Su popularidad se debe a que posibilitan trabajar con más dimensiones que las de partida sin coste adicional, gracias a que proyectan el espacio de entrada en uno de mayor dimensionalidad. De hecho, se pueden utilizar para proyectar el espacio de partida en uno de infinitas dimensiones, lo que haría que se trabaje efectivamente con infinitas dimensiones. Es interesante trabajar con más dimensiones, porque problemas irresolubles pasan a tener solución. Por ejemplo, si se tienen tres puntos no alineados en un plano 2D, es imposible hacer pasar una línea recta por los tres puntos, sin embargo, si a las entradas $\mathbf{x}$ se les aplica alguna transformación $\phi(\cdot)$ que las pase a un espacio de mayor dimensionalidad, en este nuevo espacio, un hiperplano (el equi-

valente a una línea recta en 2D), puede pasar por los tres puntos. Después solo habría que deshacer la transformación para volver al espacio original. Sin embargo, al utilizar kernels no hace falta hacer y deshacer explícitamente ninguna transformación $\phi(\cdot)$, porque ellos la hacen implícitamente debido al truco del kernel [21, Capítulo 6].

Existen muchos tipos de kernels: lineales, polinómicos, estacionarios, homogéneos, etc. Además, se pueden utilizar diversas operaciones, como las que se recogen en [21, Sección 6.2], para obtener nuevos kernels a partir de dos anteriores. En las figuras 2.1(a) y 2.1(b) el kernel que utilizaban los $\mathcal{GP}$s es el squared exponential, y viene definido por la siguiente expresión:

$$K_{SE}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\ell^2}\right\} \tag{2.7}$$

donde ℓ es el length-scale y σ_f es la varianza de la función. Estos son los hiperparámetros del kernel. Se puede apreciar que, cuanto más cerca estén dos puntos, más cercana será la norma $||\cdot||$ de su diferencia a 0, y por tanto el valor de la exponencial será más cercano a 1. Es decir, puntos muy cercanos influirán mucho entre ellos, mientras que otros muy alejados a penas se afectarán entre ellos.

El kernel utilizado en los experimentos de este trabajo ha sido Matérn52 [23], el cual resulta de sustituir ν por $5/2$ en la expresión general del kernel de Matérn:

$$K_{Matérn}(d) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}d}{\ell}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}d}{\ell}\right) \tag{2.8}$$

donde $d = ||\mathbf{x} - \mathbf{x}'||$ es la distancia Euclídea entre $\mathbf{x}$ e $\mathbf{x}'$, $\Gamma(\cdot)$ es la función Gamma, $K_\nu(\cdot)$ es la función de Bessel modificada de segunda especie [24], y ν es un parámetro que permite regular la suavidad. De hecho, los procesos Gaussianos con este kernel se pueden derivar $\lceil \nu \rceil - 1$ veces. Al realizar esta sustitución de $\nu = 5/2$, se obtiene:

$$K_{Matérn52}(d) = \sigma_f^2 \exp\left\{-\frac{\sqrt{5}d}{\ell}\right\} \left(1 + \frac{\sqrt{5}d}{\ell} + \frac{5d^2}{3\ell^2}\right). \tag{2.9}$$

En la Figura 2.2 se han generado cuatro funciones de una distribución predictiva con kernel SE y Matérn52. Se puede apreciar como las funciones generadas mediante el $\mathcal{GP}$ con kernel SE son más suaves que las generadas con el de Matérn52. Utilizar uno u otro tiene fuertes implicaciones prácticas, dado que implican una suposición sobre la forma de la función objetivo. En los problemas donde se utiliza BO, se suele utilizar el kernel de Matérn52 porque realiza menos suposiciones sobre la suavidad de la función objetivo, y se ha observado que en la práctica esto ha permitido que obtenga mejores resultados [2]. Por otro lado, se debe añadir que si se sustituyese $\nu = \infty$ en la expresión (2.9), se obtendría justamente el kernel SE. Esto se debe a que el kernel de Matérn es una generalización del SE que permite variar su suavidad.

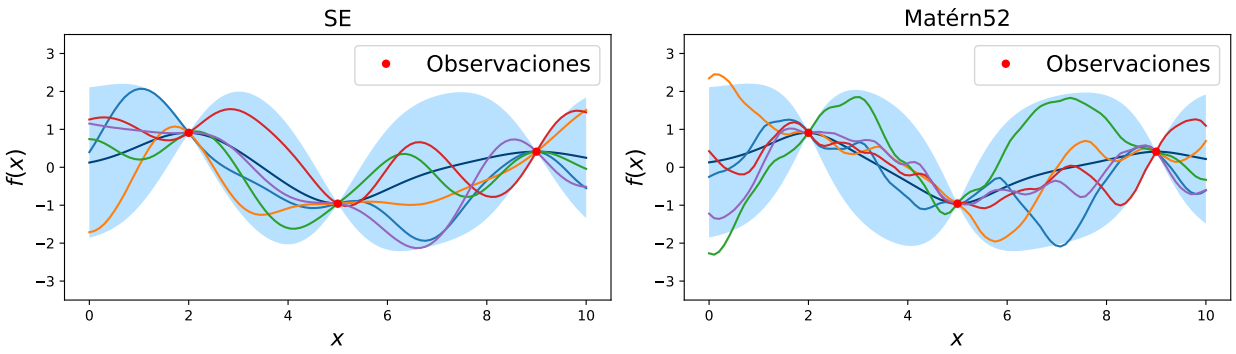


Figura 2.2: Se muestra una comparativa entre las funciones generadas mediante un $\mathcal{GP}$ a posteriori (con tres observaciones sin ruido) con kernel SE (izquierda) y otro Matérn52 (derecha), con $\ell = 1{,}0$ y $\sigma_f^2 = 1{,}0$. El área sombreada en azul claro corresponde al intervalo de confianza del 95 %, y las observaciones a las que se ha condicionado el prior se identifican con los puntos rojos.

2.2.1. Hiperparámetros del kernel

Se ha visto que tanto el kernel SE como Matérn52 tienen los hiperparámetros length-scale ℓ y la varianza de la función σ_f^2. En la Figura 2.3 se muestra una comparativa entre ambos para varios valores de sus hiperparámetros. Se puede observar que el kernel de Matérn52 es más estrecho que el SE, lo que implica que realiza menos asunciones sobre la suavidad de la función objetivo. Asimismo, se puede apreciar que σ_f^2 ha incrementado la altura de la función, este parámetro permite variar la escala pero no cambia la forma, y ℓ influye en la anchura, haciendo que tengan más o menos peso los puntos que están más alejados.

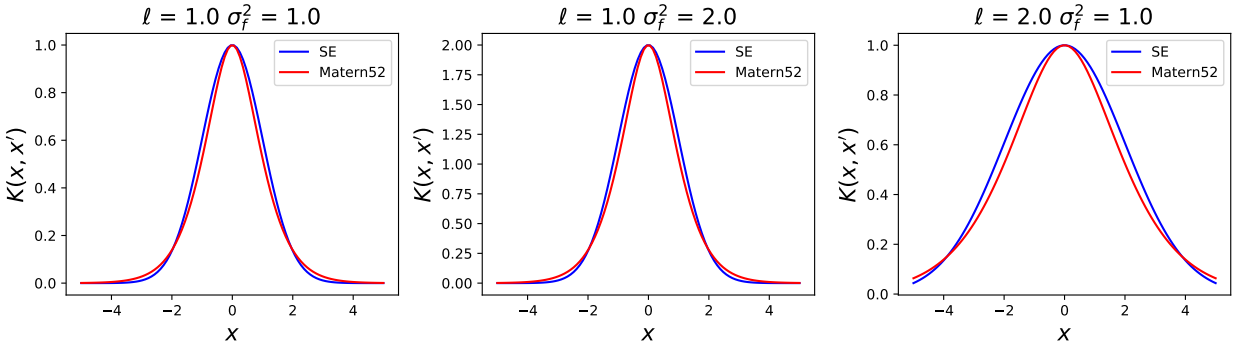
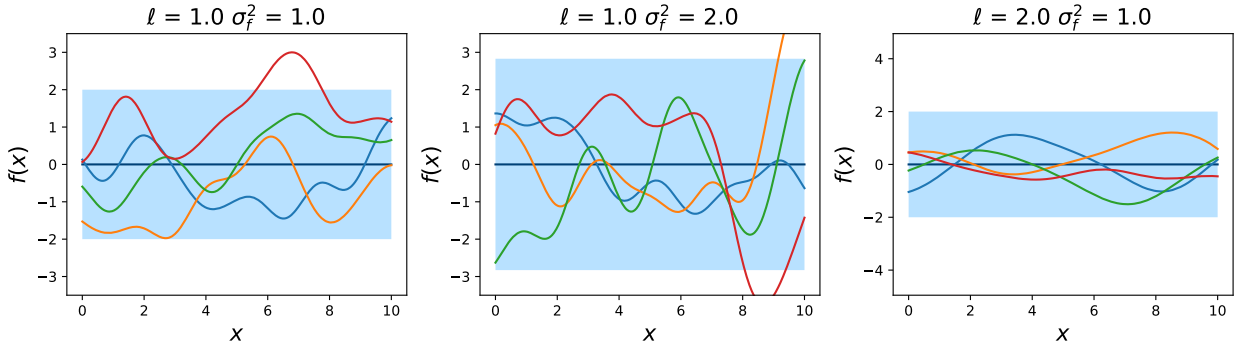


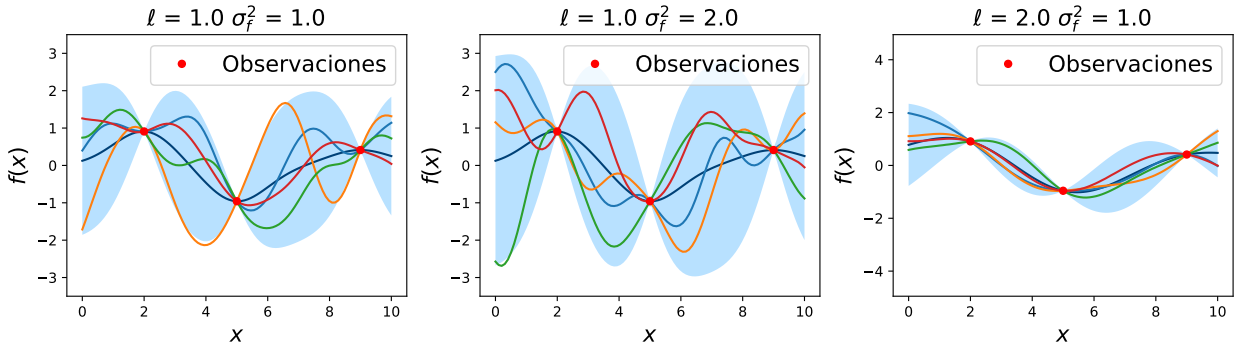
Figura 2.3: Comparativa entre el kernel SE (azul) y Matérn52 (rojo) para diferentes valores de length-scale ℓ y varianza de la función σ_f^2.

Por otro lado, en la Figura 2.4, se muestra una comparativa entre funciones generadas de un proceso Gaussiano a priori y a posteriori con diferentes valores para los hiperparámetros. Se puede ver, que el length-scale permite regular una proporción sobre cuanto puede cambiar el valor de $f(\mathbf{x})$ al modificar el valor de entrada $\mathbf{x}$. Al aumentar el length-scale se reduce la incertidumbre, y por tanto,

también disminuye la variabilidad de los lugares por los que pasa la función objetivo. Por otro lado, al aumentar σ_f^2 se especifica que la función objetivo tiene más varianza, y en consecuencia, el área del 95 % de confianza por donde pasa la función objetivo aumenta.



(a) Priori



(b) Posteriori

Figura 2.4: En las gráficas se muestran cuatro funciones generadas de un $\mathcal{GP}$ a priori (a) y otro a posteriori (b) (con tres observaciones y sin ruido) utilizando diferentes valores de length-scale y varianza. La zona azul sombreada corresponde al intervalo de confianza del 95 % del $\mathcal{GP}$, y la línea azul oscura es la media del $\mathcal{GP}$. En 2.4(b) las observaciones a las que se ha condicionado el priori son los puntos rojos.

2.3. Aprendizaje de los hiperparámetros del kernel

Cada vez que se calcula la distribución predictiva mediante el condicionamiento del priori a las observaciones, se deben ajustar los hiperparámetros del kernel del $\mathcal{GP}$. Esto se puede realizar calculando el gradiente de la verosimilitud marginal e igualando a cero. Sin embargo, este método tiende a sobreajustar (overfitting) cuando hay pocos datos [2] o muchos hiperparámetros [22, Capítulo 5], debido a que está maximizando la probabilidad de los datos. En el contexto de la optimización Bayesiana, se podría realizar otro enfoque que tuviera un tratamiento completamente Bayesiano. Sin embargo,

este requeriría marginalizar los hiperparámetros del producto de la función de adquisición por la probabilidad posterior de los hiperparámetros, es decir, habría que resolver la siguiente integral:

$$\hat{\alpha}(\mathbf{x}) = \int \alpha(\mathbf{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|X, \mathbf{y}) d\boldsymbol{\theta} \tag{2.10}$$

donde $\hat{\alpha}(\mathbf{x})$ es la función de adquisición después de marginalizar los hiperparámetros, $\alpha(\mathbf{x}|\boldsymbol{\theta})$ es la función de adquisición con los hiperparámetros $\boldsymbol{\theta}$, y el posterior de los hiperparámetros viene dado por:

$$p(\boldsymbol{\theta}|X, \mathbf{y}) \propto p(\mathbf{y}|X, \boldsymbol{\theta}) p(\boldsymbol{\theta}) p(\mathbf{y}|X) \tag{2.11}$$

donde $p(\mathbf{y}|X, \boldsymbol{\theta})$ es la probabilidad de los valores observados dados los puntos de entrada y los hiperparámetros del kernel, $p(\boldsymbol{\theta})$ es la probabilidad a priori de los hiperparámetros, y el término que normaliza es:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{2.12}$$

La integral de la ecuación (2.10) se podría aproximar por Montecarlo promediando las funciones de adquisición que surgen de las muestras de la distribución posterior de los hiperparámetros $p(\boldsymbol{\theta}|X, \mathbf{y})$. Sin embargo, $p(\boldsymbol{\theta}|X, \mathbf{y})$ es intratable debido a que no se puede resolver la integral de (2.12). A fin de generar muestras de $p(\boldsymbol{\theta}|X, \mathbf{y})$ se puede utilizar slice sampling [25]. Esta técnica ha mostrado buenos resultados, y se puede mejorar su eficiencia gracias a que se puede paralelizar [26]. Por otro lado, este es el método utilizado en el código implementado.

3

Optimización Bayesiana

3.1. Introducción a la optimización Bayesiana

En los problemas de optimización, cada vez que se realiza una evaluación, se obtiene un valor que normalmente está en el dominio de los números reales $\mathbb{R}$. Los algoritmos de optimización se pueden dividir en dos tipos [27]. Los del primer grupo se utilizan en contextos donde es barato realizar evaluaciones, tienen poco ruido, se puede obtener el gradiente con un coste adicional bajo, y la dimensionalidad puede ser muy alta. Por el contrario, los del segundo grupo se aplican cuando es muy costoso evaluar, las observaciones están contaminadas por ruido y la dimensionalidad es relativamente baja. Como se mencionó en la Sección 1.1, BO es una técnica que pertenece a este segundo grupo, motivo por el cual utiliza los valores observados para elegir dónde evaluar en la siguiente iteración de su algoritmo. Este funcionamiento permite a BO reducir la cantidad de observaciones a realizar. Además, es posible utilizar BO en múltiples problemas de diferentes ámbitos. Por ejemplo se podría utilizar para reducir la cantidad de perforaciones a realizar para encontrar petróleo, o para optimizar los parámetros de un brazo mecánico, donde cada evaluación requiere preparar un costoso experimento donde se compruebe si dicho brazo es capaz de resolver una determinada tarea [2].

Tal como se dijo en 1.1, con el propósito de lograr reducir la cantidad de evaluaciones a realizar BO utiliza un modelo probabilístico para modelar la función objetivo $f(\cdot)$ a optimizar, y una función de adquisición $\alpha(\cdot)$ cuya finalidad es medir la utilidad de evaluar en cada punto del espacio de entradas $\mathcal{X}$ en base a la información que provee el modelo probabilístico. Típicamente se utilizan los procesos Gaussianos como modelo probabilístico de la caja negra (la función objetivo), aunque es posible utilizar otros modelos, como por ejemplo, procesos t-Student o redes neuronales [28, 29]. En este trabajo se han utilizado los procesos Gaussianos, los cuales se explican en el Capítulo 2. Por otra parte, la función de adquisición tiene como objetivo guiar la búsqueda en el espacio de entradas, es decir, elegir el punto $\mathbf{x}_{N+1}$ que se espera que ayude más en la tarea de optimización. Esta función es explicada en la Sección 3.2, y el algoritmo estándar de BO se muestra en el Algoritmo 3.1. Por otro lado, se debe añadir que en los problemas de optimización el espacio de entradas $\mathcal{X}$ es acotado a una región de interés $[l_i, u_i]$, donde l_i y u_i son el límite inferior y superior de la i-ésima dimensión de entrada.

input : N número de evaluaciones a realizar,
$f(\cdot)$ caja negra a optimizar

1 **for** $n = 1, ..., N$ **do**
2 actualizar el $\mathcal{GP}$;
3 calcular $\alpha_{n-1}(\cdot)$;
4 obtener $\mathbf{x}_n$ mediante la maximización de $\alpha_{n-1}(\cdot)$: $\mathbf{x}_n = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha_{n-1}(\mathbf{x})$;
5 medir $y_n = f(\mathbf{x}_n) + \epsilon_n$;
6 guardar $(\mathbf{x}_n, y_n)$ en $\mathcal{D}_{n-1}$: $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{(\mathbf{x}_n, y_n)\}$;
7 **end**
8 **return** $\mathbf{x}_{best}$: $\arg\max_{\mathbf{x}\in\mathcal{X}} m_N(\mathbf{x})$

Algoritmo 3.1: Algoritmo de optimización Bayesiana.

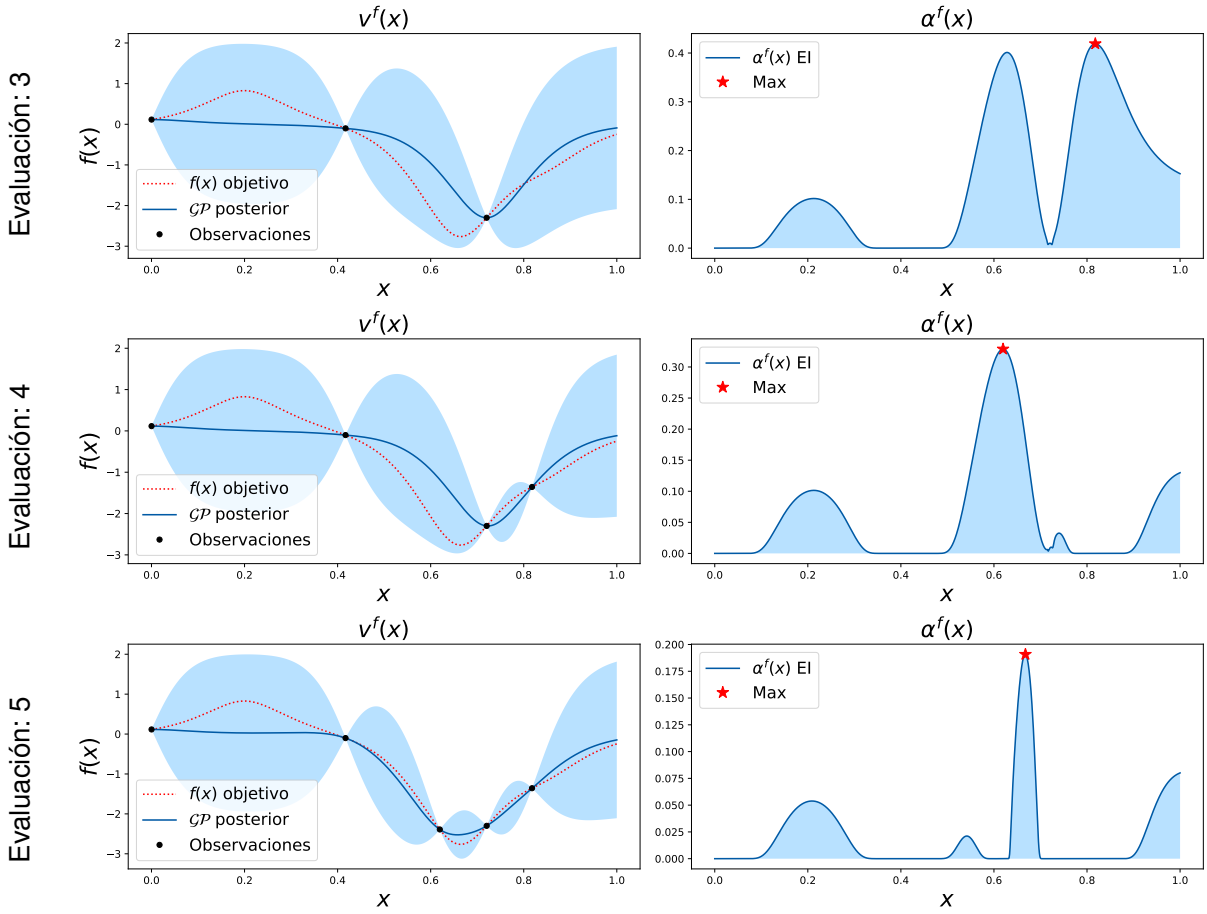


Figura 3.1: Se ilustran tres iteraciones del algoritmo de BO en un problema unidimensional. En la columna izquierda la línea roja punteada corresponde a los valores de la función objetivo, la línea azul es la media de la distribución posterior del $\mathcal{GP}$ (con kernel de Matérn52), y el área azul es el intervalo de $95\,\%$ de confianza de dicho posterior. Por otro lado, en la columna derecha se presentan los valores de la función de adquisición EI para el $\mathcal{GP}$ de la columna izquierda.

En la Figura 3.1, se presentan tres iteraciones del algoritmo de BO en un problema unidimensional, utilizando como función de adquisición EI, la cual será comentada en la Sección 3.2. Se puede ver que después de 5 observaciones la función de adquisición ya ha encontrado la región donde se sitúa el mínimo. Aunque, después de la sexta evaluación, aumentará la importancia de las zonas no exploradas, es decir, donde la varianza todavía es alta. Sin embargo, las zonas poco prometedoras según el modelo, como la que se localiza de $0{,}0$ a $0{,}4$ no serán exploradas, de modo que esta estrategia se presenta como opción superior a aquellas que realizan una exploración uniforme del espacio como la búsqueda en rejilla o la búsqueda aleatoria.

3.2. Función de adquisición en problemas de un objetivo sin restricciones

Anteriormente se ha mencionado que la función de adquisición se utiliza para guiar la búsqueda en el espacio de entradas $\mathcal{X}$. Con el propósito de que esta guía ayude a minimizar el número de observaciones a realizar, la función de adquisición debe realizar un buen balance (trade-off) entre evaluar zonas del espacio alejadas de puntos donde ya se ha evaluado (exploración), y evaluar regiones cercanas a puntos donde ya se han obtenido buenos resultados (explotación). Por un lado, se debe señalar, que como en este trabajo el objetivo es minimizar, se consideran buenas aquellas observaciones cuyos valores son bajos. Por otro lado, para realizar este balance, la función de adquisición utiliza la media y la varianza de la distribución predictiva a posteriori de los $\mathcal{GP}$. Una vez que ya se ha calculado la función de adquisición, la siguiente evaluación $\mathbf{x}_{N+1}$ simplemente es el punto donde esta sea máxima, es decir, $\mathbf{x}_{N+1} = \arg\max_{\mathbf{x}\in\mathcal{X}\subset\mathbb{R}^D} \alpha(\mathbf{x})$. Por último, es importante resaltar, que la función de adquisición no utiliza en ningún momento la verdadera función objetivo, por lo que es muy rápida de evaluar y maximizar.

Una de las primeras funciones de adquisición desarrolladas es PI, la cual utiliza la probabilidad de mejorar para guiar la búsqueda del óptimo [6]. Sin embargo, a causa de su naturaleza es pura explotación, porque cuando encuentra un punto cuyo valor es bueno respecto del resto, enfoca toda la búsqueda en alrededores de dicho punto, por lo que suele quedarse en mínimos locales [1]. Su expresión matemática es la siguiente:

$$\begin{aligned} \alpha_{\mathrm{PI}}(\mathbf{x}) &= P(f(\mathbf{x}) \leq f(\mathbf{x}^+) - \xi) \\ &= \Phi\left(\frac{f(\mathbf{x}^+) + \xi - m(\mathbf{x})}{\sqrt{v(\mathbf{x})}}\right) \end{aligned} \tag{3.1}$$

donde ξ es una constante de regularización, $f(\mathbf{x}^+)$ el mejor valor encontrado hasta el momento, $m(\mathbf{x})$ es la media de la distribución predictiva, $v(\mathbf{x})$ es la varianza de la distribución predictiva, y $\Phi(\cdot)$ es la distribución de probabilidad acumulada de una Gaussiana estándar.

Una función de adquisición que surgió con posterioridad PI que realiza un buen balance entre exploración y explotación porque guía la búsqueda en función de la mejora esperada es EI [7]. Gracias al buen balance que realiza ha obtenido buenos resultados en diversos problemas [2]. La definición de EI es la siguiente:

$$\alpha_{\mathrm{EI}}(\mathbf{x}) = \begin{cases} (f(\mathbf{x}^+) + \xi - m(\mathbf{x}))\Phi(\zeta) + \sqrt{v(\mathbf{x})}\phi(\zeta) & \text{si } \sqrt{v(\mathbf{x})} > 0 \\ 0 & \text{si } \sqrt{v(\mathbf{x})} = 0 \end{cases} \tag{3.2}$$

donde $\phi(\cdot)$ es la función de densidad de probabilidad de una Gaussiana estándar, y

$$\zeta = \begin{cases} \frac{f(\mathbf{x}^+)+\xi-m(\mathbf{x})}{\sqrt{v(\mathbf{x})}} & \text{si } \sqrt{v(\mathbf{x})} > 0 \\ 0 & \text{si } \sqrt{v(\mathbf{x})} = 0. \end{cases}$$

Sus buenos resultados han originado que se haya adaptado a otros problemas en los que no solo hay una función objetivo. Algunas de sus adaptaciones serán comentadas en el Capítulo 4.

Una función de adquisición más sencilla que PI y EI, es LCB, la cual confía en el límite de menor confianza para guiar la búsqueda [8]. Su expresión es la siguiente:

$$\alpha_{\mathrm{LCB}}(\mathbf{x}) = m(\mathbf{x}) - \kappa\sqrt{v(\mathbf{x})} \tag{3.3}$$

donde κ es una constante a establecer por quién utilice esta función de adquisición. Esta función tiende a confiar demasiado en la varianza de la distribución predictiva, aunque sus resultados suelen ser mejores a los de PI [2].

Por otro lado, recientemente surgió una nueva función de adquisición que a diferencia con las anteriores basa su búsqueda en la teoría de la información. El nombre de esta novedosa función de adquisición es búsqueda de la entropía (ES). Esta función de adquisición confía en la reducción de la entropía de la solución para guiar la búsqueda del óptimo [27]. Su expresión es la siguiente:

$$\alpha_{\mathrm{ES}}(\mathbf{x}) = H\left[p(\mathbf{x}^\star|\mathcal{D})\right] - \mathbb{E}_{p(y|\mathcal{D},\mathbf{x})}\left[H\left[p(\mathbf{x}^\star|\mathcal{D}\cup\{(\mathbf{x},y)\})\right]\right] \tag{3.4}$$

donde $\mathbf{x}^\star$ es la localización del máximo, $H\left[p(\mathbf{x}^\star|\mathcal{D})\right]$ es la entropía de $\mathbf{x}^\star$ dada por el modelo probabilístico actual, $H\left[p(\mathbf{x}^\star|\mathcal{D}\cup\{(\mathbf{x},y)\})\right]$ es la entropía de $\mathbf{x}^\star$ después de añadir la nueva observación $(\mathbf{x},\mathbf{y})$ al conjunto de datos $\mathcal{D}$, y la esperanza es respecto de los valores de $\mathbf{y}$ dado el conjunto de datos $\mathcal{D}$ y el punto donde se evalúa $\mathbf{x}$. Debido a sus buenos resultados aparecieron otras funciones de adquisición que también confían en la minimización de la entropía (es decir, maximización de la información) de la solución del problema pero cuya formulación es distinta para evitar las costosas aproximaciones de ES. La primera de ellas es búsqueda de la entropía predictiva (PES), cuya expresión es:

$$\alpha_{\mathrm{PES}}(\mathbf{x}) = H\left[p(y|\mathcal{D},\mathbf{x})\right] - \mathbb{E}_{p(\mathbf{x}^\star|\mathcal{D})}\left[H\left[p(y|\mathcal{D},\mathbf{x},\mathbf{x}^\star)\right]\right] \tag{3.5}$$

donde $p(y|\mathcal{D},\mathbf{x},\mathbf{x}^\star)$ es la distribución posterior predictiva de y dado el conjunto de datos $\mathcal{D}$, el punto

donde se evalúa $\mathbf{x}$ y la localización del máximo $\mathbf{x}^\star$, y ahora la esperanza es respecto de las posibles localizaciones del máximo $\mathbf{x}^\star$ dado el conjunto de datos $\mathcal{D}$. Aunque la ejecución de PES es menos costosa que la de ES, su cálculo también implica costosas aproximaciones. La segunda de las funciones de adquisición basada en la teoría de la información surgió para evitar estos costosos cálculos. Aunque también requiere de ciertas aproximaciones que causan que sea más costosa que funciones de adquisición tradicionales como PI, EI o LCB. Esta función de adquisición parte de un punto de vista diferente al de ES y PES que consiste en realizar una búsqueda basada la entropía del máximo valor en lugar de una búsqueda basada en la localización del máximo. Esta nueva perspectiva da nombre a esta función de adquisición, la cual se llama búsqueda de la entropía del máximo valor (MES). Su expresión es la siguiente:

$$\alpha_{\mathrm{MES}}(\mathbf{x}) = H\left[p(y|\mathcal{D},\mathbf{x})\right] - \mathbb{E}_{p(y^\star|\mathcal{D})}\left[H\left[p(y|\mathcal{D},\mathbf{x},y^\star)\right]\right] \tag{3.6}$$

donde se puede ver que el primer término es exactamente el mismo que el de la expresión (3.5) de PES, el segundo también es muy similar al de PES pero en lugar de utilizar la localización del máximo $\mathbf{x}^\star$, usa el valor del máximo $y^\star$, y ahora la esperanza es respecto de los posibles valores del máximo $y^\star$ dado el conjunto de datos $\mathcal{D}$. Esta última función de adquisición (MES) es la empleada para crear MESMOC, una función de adquisición que puede abordar problemas que no solo tienen una función objetivo.

4

Optimización Bayesiana con múltiples objetivos y restricciones desconocidas

4.1. Introducción a la optimización Bayesiana con múltiples objetivos y restricciones desconocidas

En los problemas de optimización con múltiples objetivos no suele existir una única solución óptima al problema. Debido a que una configuración de parámetros $\mathbf{x}$ puede dar buenos resultados en la función objetivo $f_1(\mathbf{x})$ a cambio de obtener un desempeño bastante pobre en $f_2(\mathbf{x})$, o por el contrario dar muy buenos resultados en $f_2(\mathbf{x})$ debido a que su actuación en $f_1(\mathbf{x})$ es mediocre. Por ejemplo, si se quisiera maximizar la velocidad de un robot y también minimizar su consumo energético, como ambos objetivos son conflictivos, no hay una solución que obtenga el mejor resultado en ambos, dado que aumentar la velocidad requerirá de más energía y minimizar el consumo reducirá la velocidad. De hecho, existe una cantidad infinita de configuraciones óptimas, estas conforman el conjunto de Pareto $\mathcal{X}^\star$ [3]. Asimismo, los valores de salida de estas configuraciones forman la frontera de Pareto $\mathcal{Y}^\star$. Las configuraciones del conjunto de Pareto $\mathcal{X}^\star$ son óptimas porque no son Pareto-dominadas por ningún otro punto, es decir, no hay puntos que sean mejores en todas las funciones objetivo a los puntos que conforman $\mathcal{X}^\star$. En función del contexto de optimización, ser mejor significa ser menor, si se está minimizando, o ser mayor, si se está maximizando. Es importante apreciar que el tamaño de $\mathcal{X}^\star$, y por tanto también el de $\mathcal{Y}^\star$ es infinito, dado que, el valor de $f(\mathbf{x})$ para una entrada cualquiera $\mathbf{x} \in \mathbb{R}^D$, con D la dimensión de entrada, es un valor en $\mathbb{R}$.

En los problemas abordados en este trabajo el objetivo es minimizar, por tanto, como se comentó anteriormente, el valor de $f(\mathbf{x})$ es superior al de $f(\mathbf{x}')$ si $f(\mathbf{x}) < f(\mathbf{x}')$. Al llevar esto al contexto donde se dispone de varios objetivos, un punto Pareto-domina a otro si los valores que obtiene en todas las funciones objetivo son menores o iguales que los obtenidos por el otro, excepto un valor que debe ser estrictamente menor. En este trabajo se consideran K funciones objetivo, así que, $\mathbf{x}_1$ domina a $\mathbf{x}_2$ si $f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2)$, $\forall k = \{1, ..., K\}$, con una desigualdad estricta. Además, también se cuenta con C restricciones, las cuales se deben satisfacer para decir que un punto $\mathbf{x}$ pertenece al conjunto de puntos válidos $\mathcal{F}$. Es decir, para que un punto $\mathbf{x}$ pertenezca a $\mathcal{F}$, debe cumplir que $c_j(\mathbf{x}) \geq 0$, $\forall j = \{1, ..., C\}$. Por lo tanto, $\mathcal{F}$ es un subconjunto dentro del conjunto de posibles valores de entrada $\mathcal{X}$.

En la Figura 4.1 se muestra un ejemplo de frontera de Pareto conformada solamente por cinco puntos. Cada uno de estos puntos es un óptimo de Pareto dado que ninguno es mejor que el resto en todas las funciones objetivo. El área de puntos que domina cada uno de estos óptimos corresponde a la zona sombreada del mismo color que tiene cada punto, pero en un tono un poco más suave. El área sombreada total es el hipervolumen solución. Es fácil ver que cuanto mayor sea este hipervolumen mejor será la solución del problema, por lo que es una buena medida del desempeño que tiene una estrategia de optimización en problemas multi-objetivo. Por otro lado, en la Figura 4.2 se muestra una frontera de Pareto en el espacio funcional sin restricciones y otra con restricciones.

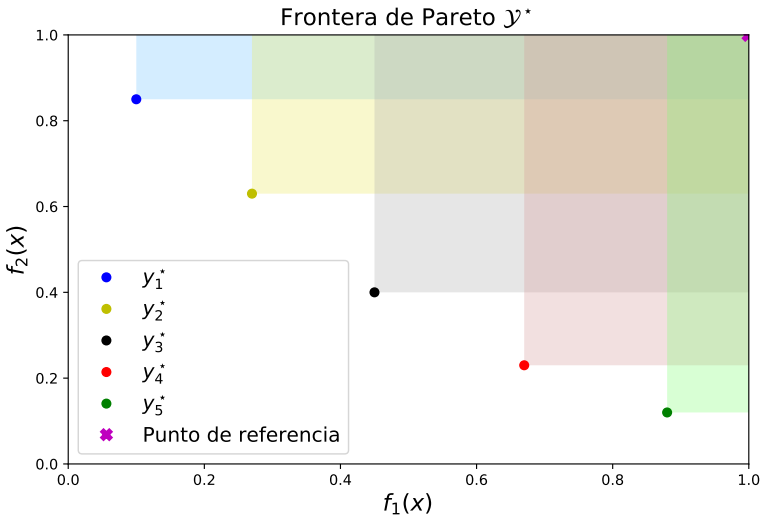
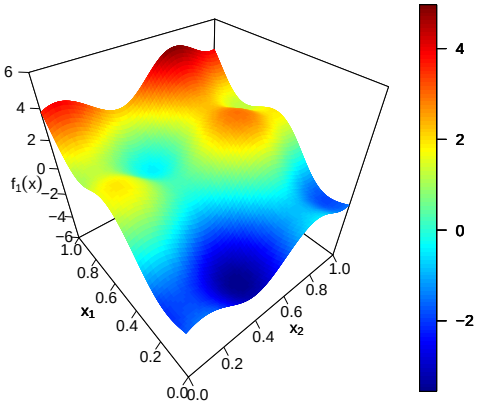
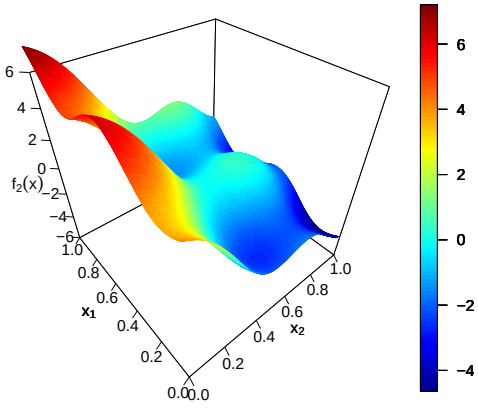


Figura 4.1: Se muestra un ejemplo de frontera de Pareto formada únicamente por 5 puntos (en diferentes colores), los cuales son Pareto-óptimos. La región de valores que domina cada punto de la frontera es un área sombreada del mismo color al que pertenece cada punto pero en un tono más claro. La región sombreada total corresponde al hipervolumen del conjunto solución actual. Para calcular este hipervolumen se utiliza un punto de referencia, el cual se sitúa en la parte superior derecha de la figura.

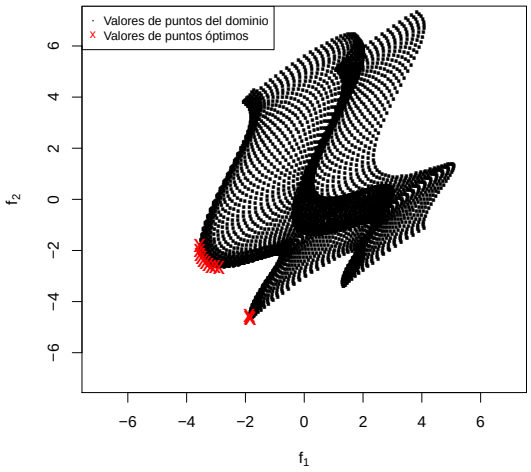
En la Sección 1.1, y en el Capítulo 3, se muestra que la optimización Bayesiana (BO) se emplea para resolver problemas donde: no se dispone de la expresión analítica de la función objetivo, realizar evaluaciones es muy costoso, y además, los valores medidos pueden estar contaminados por ruido. Para resolver problemas con estas características, la BO en problemas multi-objetivo con restricciones construye un modelo probabilístico por cada función objetivo $f_k(\cdot)$ y cada restricción $c_j(\cdot)$, y utiliza la función de adquisición $\alpha(\cdot)$ para medir la utilidad esperada de evaluar en cada punto. Dada la naturaleza de los problemas donde se aplica BO, lo ideal es realizar el menor número de evaluaciones y aprovechar al máximo los modelos de los objetivos y las restricciones. Motivo por el cual, BO se presenta como una estrategia superior a otros posibles métodos de elección de nuevas evaluaciones como búsqueda en rejilla, búsqueda aleatoria o utilizar algoritmos evolutivos, dado que las dos primeras no utilizan el modelo para guiar la búsqueda y la tercera necesitaría de muchas evaluaciones.



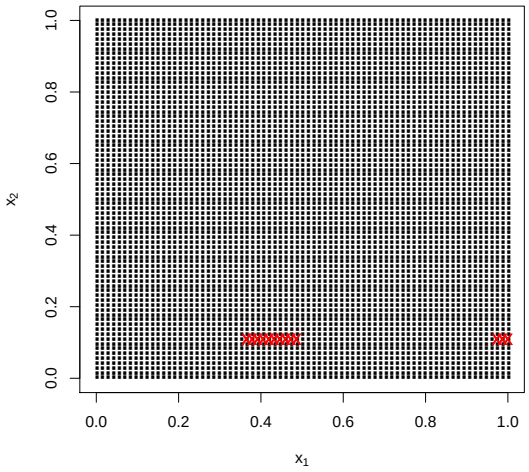
(a) Objetivo 1



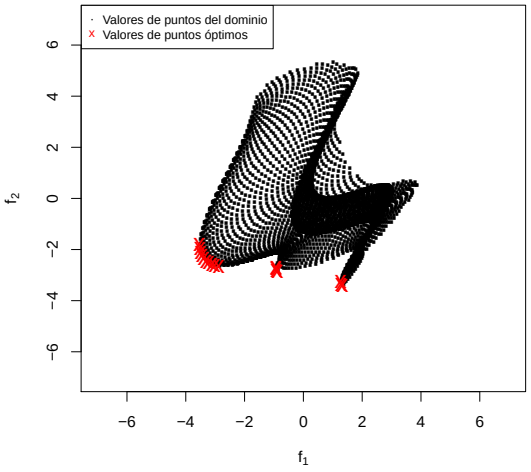
(b) Objetivo 2



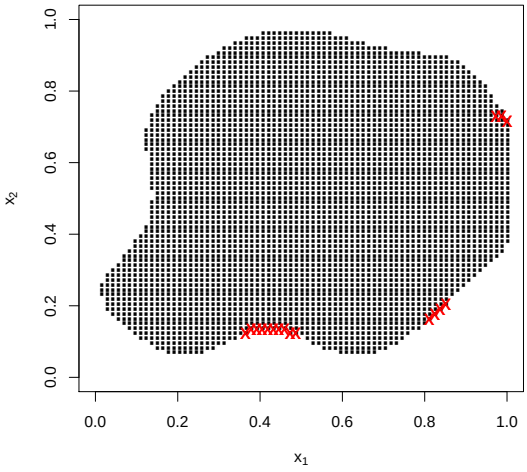
(c) Frontera de Pareto



(d) Conjunto de Pareto



(e) Frontera de de Pareto al aplicar una restricción



(f) Conjunto de Pareto al aplicar una restricción

Figura 4.2: Se presentan dos objetivos y los puntos del conjunto de Pareto solución en el espacio funcional (la frontera de Pareto) y en el espacio de entrada (el conjunto de Pareto), sin aplicar y aplicando una restricción. También se muestra como queda la frontera y el conjunto de Pareto al prohibir los valores que pueden tomar algunos puntos de entrada. En las Subfiguras 4.2(c), 4.2(d), 4.2(e) y 4.2(f) los puntos negros corresponden a los valores de los diferentes objetivos para diferentes puntos del espacio de entradas. Por otro lado, las cruces rojas son los valores Pareto-óptimos, debido a que ninguno es superado en ambos objetivos por ningún otro punto.

En las siguientes secciones de este capítulo, se comentarán funciones de adquisición de problemas con múltiples objetivos sin restricciones, un objetivo y varias restricciones y múltiples objetivos y varias restricciones. Posteriormente se hablará de la función de adquisición PESMOC y finalmente se realizará el desarrollo de MESMOC, la función de adquisición desarrollada y probada en el presente trabajo.

4.2. Función de adquisición

Tal como se ha visto en la Sección 3.2, la función de adquisición es la segunda pieza clave de la optimización Bayesiana y tiene como finalidad regular el balance (trade-off) entre la explotación de regiones del espacio de puntos cercanos a otros ya evaluados donde se han obtenido buenos resultados, y la exploración de zonas alejadas de puntos donde ya se ha evaluado. Para ello calcula la utilidad de evaluar en cada uno de los puntos del espacio de entradas $\mathcal{X}$ (la función de adquisición es simplemente la esperanza de esa utilidad). En los problemas donde no hay solamente una caja negra, que corresponde a una función objetivo, sino que hay varias, ya sean de objetivos o restricciones, la función de adquisición utiliza las distribuciones predictivas a posteriori de los $\mathcal{GP}$ de todas estas cajas negras. Como se utilizan las distribuciones predictivas y no los verdaderos objetivos y restricciones, el coste de calcular y maximizar la función de adquisición se considera despreciable en comparación con evaluar en los verdaderos objetivos y restricciones.

4.2.1. Función de adquisición en problemas multi-objetivo

Una de las funciones de adquisición más sencillas desarrollada para afrontar problemas multi-objetivo es ParEGO [30]. Esta función es una adaptación de EI. En cada iteración ParEGO genera un vector de K pesos aleatorios de una distribución uniforme, el cual se utiliza para ponderar la importancia de cada objetivo, y así combinar todas las funciones en una sola. Posteriormente, utiliza EI para evaluar esta nueva función objetivo. Aunque ParEGO es bastante rápida, muchas otras funciones más avanzadas que no convierten el problema en uno de un solo objetivo la superan.

Anteriormente, en la Sección 4.1, se ha explicado que el hipervolumen de la frontera de Pareto permite medir la calidad de las soluciones de una estrategia de optimización. En SMSego, una función de adquisición basada en LCB, se calcula el hipervolumen de un conjunto de Pareto $\tilde{\mathcal{X}}^{\star}$ que se construye en cada iteración [11]. Para elaborarlo, se eligen las entradas $\mathbf{x}$ en base a una estimación optimista de sus valores de salida dada por $m_k(\mathbf{x}) - \kappa \cdot v_k(\mathbf{x})^{1/2}$, donde κ es una constante y m_k y v_k son la media y la varianza de la distribución predictiva, del k-ésimo objetivo, dadas por las ecuaciones (2.6a) y (2.6b) respectivamente. Después se calcula el hipervolumen de esos puntos seleccionados, se les aplica una corrección, y entonces, se elige aquella nueva observación que maximice el hipervolumen.

Otras funciones de adquisición que también trabajan naturalmente en problemas multi-objetivo son SUR y EHI, adaptaciones de PI y EI respectivamente [9, 10]. Estas funciones también construyen un conjunto de Pareto $\tilde{\mathcal{X}}^\star$ y se enfocan en maximizar el hipervolumen de su frontera $\tilde{\mathcal{Y}}^\star$. Sin embargo, esta maximización es muy costosa de realizar debido a que su coste es exponencial respecto del número de objetivos, así que los métodos que la utilizan no pueden trabajar con más de dos o tres objetivos. Por otra parte, PESMO [15] y MESMO [19] las adaptaciones para varios objetivos de PES y MES, respectivamente, son capaces de obtener resultados superiores a los de las anteriores funciones, y además, trabajar con más de dos o tres objetivos debido a que escalan mejor con el número de objetivos.

4.2.2. Función de adquisición en problemas de un objetivo con restricciones

Gran parte de las métodos utilizados para afrontar el problema de un objetivo y varias restricciones son extensiones de EI [7, 31]. Se pueden añadir las restricciones a EI simplemente multiplicando su función de adquisición por la probabilidad de que se cumplan las restricciones, para así obtener EIC [12–14]. Es decir, $\alpha_{\mathrm{EIC}(\mathbf{x})} = \alpha_{\mathrm{EI}}(\mathbf{x}|\eta, \mathcal{D}^f)h(\mathbf{x})$, donde η es el mejor valor obtenido que cumple las restricciones, $\mathcal{D}^f$ es el conjunto de datos de entrada con los valores obtenidos en la función objetivo (es decir, no tiene los valores de las restricciones), y $h(\mathbf{x})$ es la probabilidad conjunta de cumplir todas las restricciones dado $\mathcal{D}^c$, el cual es el conjunto de datos de entrada con los valores obtenidos en todas las restricciones. Sin embargo, cuando todavía no se tiene ningún punto válido, no existe η, y por tanto, no se puede calcular $\alpha_{\mathrm{EI}}(\mathbf{x}|\eta, \mathcal{D}^f)$. Una posible solución consiste en utilizar una función de adquisición diferente cuando todavía no se han obtenido puntos válidos. Esta función solamente se enfocaría en encontrar valores que satisfagan las restricciones [13]. No obstante, esta solución no es la mejor posible dado que la función de adquisición debería contemplar directamente un escenario donde todavía no haya observaciones válidas. Por otro lado, las extensiones de las funciones de adquisición PES y MES para problemas con restricciones PESC [16] y MESC [20], respectivamente, no tienen esta dificultad, y además, obtienen unos resultados semejantes o superiores a los de EIC.

4.2.3. Función de adquisición en problemas multi-objetivo con varias restricciones

A fin de utilizar una función de adquisición que sea capaz de abordar problemas multi-objetivo sujetos a varias restricciones, se deben aplicar las adaptaciones que se hacían para trabajar con múltiples objetivos, y también las que se usaban para trabajar con varias restricciones. Sin embargo, normalmente, esto no se puede hacer de manera directa, dado que ambas modificaciones deben ser compatibles entre ellas.

BMOO es una función de adquisición capaz de abordar problemas con múltiples objetivos y restricciones. Se obtiene BMOO al multiplicar un EHI modificado por la probabilidad conjunta de satisfacer todas las restricciones [4]. Una de las modificaciones de este EHI (respecto del EHI original), surge del hecho de que ahora puede no haberse obtenido hasta el momento ningún punto válido, es decir, $\nexists \mathbf{x}$ tal que $\mathbf{x} \in \mathcal{F}$. Para resolver esto, se le debe dar más importancia a que aquellos puntos que están más cerca de pertenecer a $\mathcal{F}$. Para ello, se aplica una transformación a los valores de salida de las funciones objetivo y las restricciones, lo que provoca que la regla de dominación de Pareto cambie. La transformación a realizar es la siguiente:

$$\Psi(\mathbf{y}^f, \mathbf{y}^c) = \begin{cases} (\mathbf{y}^f, \mathbf{0}) & si \quad \mathbf{y}^c \geq 0 \\ (+\infty, \min(\mathbf{y}^c, \mathbf{0})) & en\ otro\ caso \end{cases}$$

donde

$$\mathbf{y}^f = \{y_1^f, ..., y_K^f\} = \{y_1^f(\mathbf{x}), ..., y_K^f(\mathbf{x})\} = \{f_1(\mathbf{x}) + \epsilon_1^f, ..., f_K(\mathbf{x}) + \epsilon_K^f\}$$
$$\mathbf{y}^c = \{y_1^c, ..., y_C^c\} = \{y_1^c(\mathbf{x}), ..., y_C^c(\mathbf{x})\} = \{c_1(\mathbf{x}) + \epsilon_1^c, ..., c_C(\mathbf{x}) + \epsilon_C^c\}$$

donde el ruido de los objetivos y las restricciones, ϵ^f y ϵ^c ha sido generado de forma independiente de una distribución normal con media 0 y varianza $\left(\sigma^f\right)^2$ y $(\sigma^c)^2$, respectivamente. Las propiedades de la regla de dominación de Pareto dada esta transformación, la cual contempla el grado de invalidez de los puntos, son las siguientes:

1.– En los problemas sin restricciones, la regla de dominación se reduce a la regla de dominación de Pareto clásica.

2.– Las soluciones válidas (las que cumplen que $\mathbf{y}^c \geq \mathbf{0}$) son comparadas por la regla de dominación de Pareto en el espacio de valores de las funciones objetivo, es decir, en el espacio funcional.

3.– Las soluciones inválidas (en las que $\mathbf{y}^c \ngeq \mathbf{0}$) son comparadas utilizando la regla de dominación de Pareto aplicada al vector de violación de restricciones.

4.– Las soluciones válidas siempre dominan a las inválidas.

La función de adquisición de BMOO que resulta es la siguiente:

$$\alpha(\mathbf{x}) = \mathbb{E}_{\mathbf{y}^f, \mathbf{y}^c}\left[\int_{\mathcal{G}_N} \mathbb{I}(\Psi(\mathbf{y}^f, \mathbf{y}^c) \lhd \mathbf{y}) d\mathbf{y}\right] = \int_{\mathcal{G}_N} p(\Psi(\mathbf{y}^f, \mathbf{y}^c) \lhd \mathbf{y}) d\mathbf{y} \tag{4.1}$$

donde $\mathbf{a} \lhd \mathbf{b}$ significa que $\mathbf{a}$ es Pareto dominado por $\mathbf{b}$, $\mathbb{I}(\cdot)$ es la función indicatriz, y $\mathcal{G}_N$ es el conjunto de puntos no dominados hasta la iteración N (por lo que $\tilde{\mathcal{X}}^\star \subset \mathcal{G}_N$). La integral de (4.1) no se puede calcular de forma analítica, así que en [4] se propone aproximarla por Montecarlo. Se debe tener en cuenta que no es trivial generar uniformemente valores aleatorios de $\mathcal{G}_N$, dado que estos deben cumplir las restricciones. Además, si se decidiera generar valores, y después aceptarlos o rechazarlos en función de si son válidos o no, el proceso sería muy costoso, y se volvería prohibitivo según aumentase el número de restricciones. En su lugar, en [4] se desarrolló una variante de una técnica llamada subset simulation para generar las muestras de Montecarlo.

Por otro lado, en [5] se elaboró PESMOC extendiendo el trabajo ya realizado para PESMO y PESC

de [15] y [16]. Esta función de adquisición ha obtenido mejores resultados a los de BMOO, además, de necesitar menos tiempo en cada iteración [5]. Asimismo, esta función cuenta con una versión desacoplada que obtiene unos resultados superiores a los de la variante acoplada. En la Sección 4.3 se explicará más en detalle como funciona este método y en la siguiente sección se discutirá sobre la función de adquisición en el entorno desacoplado.

4.2.4. Función de adquisición desacoplada

Hasta el momento, la evaluación de las cajas negras se ha comentado siempre de forma acoplada, es decir, se evalúa al mismo tiempo en todas las cajas negras y se calcula la función de adquisición utilizando los valores de los $\mathcal{GP}$ actualizados. Sin embargo, en los problemas donde hay varias cajas negras, es decir, varios objetivos o restricciones, no es necesario evaluarlas al mismo tiempo ni en el mismo punto. Es decir, cuando hay varias cajas negras surge la posibilidad de realizar evaluaciones de forma desacoplada. Aunque el entorno desacoplado implica que la función de adquisición debe estar definida como una combinación lineal del $\mathcal{GP}$ de cada caja negra, ya que debe ser posible separar una función de adquisición independiente por cada caja negra para saber la utilidad de evaluar por separado en ella. En caso de que esto no se cumpla, habría que continuar evaluando todas las cajas negras de forma acoplada. Un ejemplo donde se podría utilizar este modo desacoplado se puede ver en el problema de optimización de una red neuronal de la Sección 1.1. En este ejemplo, se debe evaluar el porcentaje de error en las predicciones y la velocidad de predicción. No obstante, la evaluación de cada una de estas cajas negras es completamente independiente de la otra, ya que para conocer la velocidad de predicción únicamente es necesario crear una red con pesos aleatorios y cronometrar el tiempo que necesita para predecir, pero no es necesario saber su error, mientras que para calcular el porcentaje de error es necesario entrenar una red y después medir su error en un conjunto de validación, pero no se necesita saber su velocidad de predicción. Por lo tanto, dada la naturaleza de esta cajas negras, este problema permite un entorno desacoplado. En la Figura 4.3 se muestra una representación gráfica de un escenario donde las evaluaciones se realizan de forma acoplada y otro donde son de forma desacoplada. Se puede apreciar que en el escenario acoplado se evalúa en el mismo punto las dos cajas negras, mientras que en el escenario desacoplado cada caja negra se evalúa en un punto distinto.

El entorno desacoplado tiene dos tipos de funcionamiento [16], el competitivo y el no competitivo. En el entorno competitivo, en cada iteración solamente se evalúa la caja negra que más ayude en la tarea de optimización, por lo que las cajas negras compiten por ser evaluadas. Por otro lado, en el entorno no competitivo, en cada iteración se evalúan todas las cajas negras en el punto que más beneficie cada una a la optimización, así que, como se evalúan todas las cajas negras, estas no compiten entre ellas.

Cuando se utiliza un entorno desacoplado competitivo, se permite que el algoritmo de BO se centre en los objetivos o las restricciones que sean más desafiantes. Debido a ello, solamente se ha visto que

en este entorno se obtengan unos resultados superiores a los que se obtienen al evaluar de forma acoplada [5].

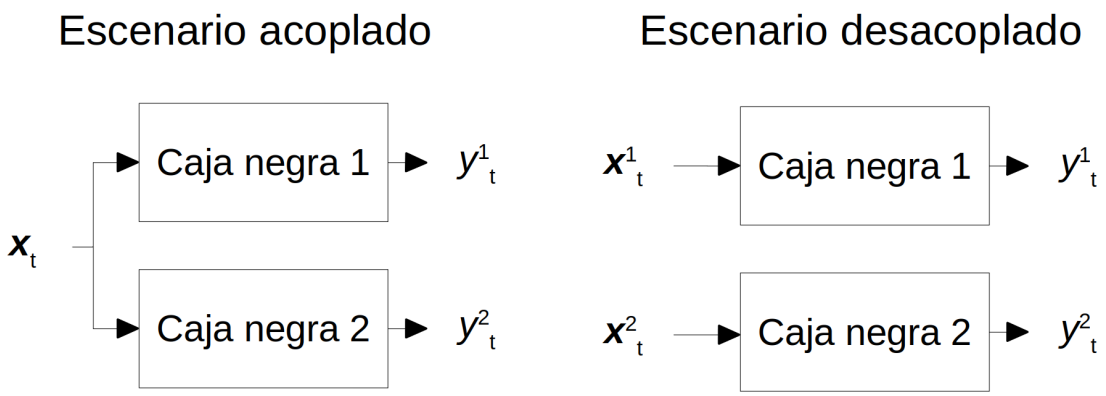


Figura 4.3: Representación gráfica de un escenario acoplado y otro desacoplado. En el escenario acoplado se evalúa en el punto $\mathbf{x}_t$ en el tiempo t, y en el escenario desacoplado se evalúa en el punto $\mathbf{x}_t^1$ en la caja negra 1 y $\mathbf{x}_t^2$ en la caja negra 2 en el tiempo t

Ninguna las funciones de adquisición multi-objetivo explicadas en la Sección 4.2.1 están preparadas para trabajar de forma desacoplada (exceptuando PESMO y MESMO). Por otro lado, EIC, y las funciones que se basan en ella, sufren una patología que las impide funcionar en un entorno desacoplado, la cual fue identificada en [13]. EIC requiere que en cada iteración se cumplan dos condiciones. La primera es que, el siguiente valor a observar debe poder ser mejor que el anterior mejor valor. La segunda es que se debe obtener un valor válido, es decir, que cumpla las restricciones. Satisfacer ambas condiciones en la misma iteración evaluando en puntos diferentes o solo evaluando una $f(\cdot)$ o una $c(\cdot)$ correspondiente, no es posible. Por lo que EIC no se puede utilizar de forma desacoplada.

Las funciones de adquisición citadas que se basan en la teoría de la información tienen una forma que las predispone para ser utilizadas de forma no acoplada. Esto se debe a que codifican la adquisición total como una suma de la adquisición de los objetivos y las restricciones. Por lo que al tener la adquisición de cada f y c, es posible que compitan entre ellas para decidir en qué objetivo o restricción evaluar.

4.3. Búsqueda de entropía predictiva con múltiples objetivos y restricciones desconocidas

La explicación de esta sección se realizará siguiendo [5], trabajo en el cual se desarrolla PESMOC. Además, la descripción que se efectuará será para la versión acoplada. Sin embargo, el cambio que hay que llevar a cabo para que PESMOC sugiera evaluaciones de forma desacoplada es muy sencillo, si se parte de su variante acoplada, y será tratado al final de la Sección 4.3.2.

Desde aquí en adelante, $\mathbf{f}$ denotará el conjunto de K funciones objetivo $\{f_1, ..., f_K\}$ y $\mathbf{c}$ será el conjunto de C restricciones $\{c_1, ..., c_C\}$. Además, en lugar de utilizar dos vectores $\mathbf{y}^f$ y $\mathbf{y}^c$ para las salidas de las funciones objetivo y restricciones a las que ya se les ha añadido el ruido, se utilizará un único vector $\mathbf{y}$ de tamaño $K + C$ (donde $\mathbf{y} = \{(\mathbf{y}^f, \mathbf{y}^c)\}$). Así que, el conjunto de datos será $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, donde N es el número de iteraciones del algoritmo realizadas hasta el momento.

La función de adquisición PESMOC se puede utilizar en problemas con múltiples objetivos y con varias restricciones. En estos problemas la solución es un conjunto de Pareto $\mathcal{X}^\star$ cuyos valores pertenecen a $\mathcal{F}$, así que, $\mathcal{X}^\star \subset \mathcal{F}$. Además, PESMOC pertenece al conjunto de funciones basadas en minimización de la entropía de la solución del problema. Por lo que el punto $\mathbf{x}_{N+1}$ donde recomienda evaluar en la siguiente iteración, es aquel, donde más se reduce la entropía de la distribución predictiva del conjunto de Pareto, es decir, la entropía de $p(\mathcal{X}^\star|\mathcal{D})$, después de conocer su valor. Esto se recoge en la siguiente expresión:

$$\alpha(\mathbf{x}) = H\left[p(\mathcal{X}^\star|\mathcal{D})\right] - \mathbb{E}_{p(\mathbf{y}|\mathcal{D},\mathbf{x})}\left[H\left[p(\mathcal{X}^\star|\mathcal{D} \cup \{(\mathbf{x}, \mathbf{y})\})\right]\right] \tag{4.2}$$

donde $H\left[p(\mathcal{X}^\star|\mathcal{D})\right]$ es la entropía de $\mathcal{X}^\star$ dada por los modelos probabilísticos de los objetivos y las restricciones actuales, $H\left[\mathcal{X}^\star|\mathcal{D} \cup \{(\mathbf{x}, \mathbf{y})\}\right]$ es la entropía de $\mathcal{X}^\star$ después de actualizar las distribuciones predictivas por el nuevo dato $(\mathbf{x}, \mathbf{y})$, y la esperanza $\mathbb{E}[\cdot]$ es sobre la probabilidad conjunta de los objetivos y las restricciones añadiéndoles el ruido, es decir, los posibles valores de $\mathbf{y}$, dado el conjunto de datos $\mathcal{D}$ y el punto donde se evalúa $\mathbf{x}$.

Sin embargo, la expresión (4.2) es intratable, porque habría que calcular la entropía del conjunto $\mathcal{X}^\star$, el cual tiene un tamaño potencialmente infinito. Esto se puede resolver al darse cuenta de que (4.2) es la información mutua entre $\mathcal{X}^\star$ e $\mathbf{y}$, y como la información mutua es simétrica ($I(\mathcal{X}^\star, \mathbf{y}) = I(\mathbf{y}, \mathcal{X}^\star)$), es posible reescribir esta expresión intercambiando $\mathcal{X}^\star$ e $\mathbf{y}$. De forma que, se obtiene lo siguiente:

$$\alpha(\mathbf{x}) = H\left[p(\mathbf{y}|\mathcal{D}, \mathbf{x})\right] - \mathbb{E}_{p(\mathcal{X}^\star|\mathcal{D})}\left[H\left[p(\mathbf{y}|\mathcal{D}, \mathbf{x}, \mathcal{X}^\star)\right]\right] \tag{4.3}$$

donde $H\left[p(\mathbf{y}|\mathcal{D}, \mathbf{x})\right]$ es la entropía de las distribuciones predictivas de los objetivos y las restricciones dado $\mathcal{D}$, $H\left[\mathbf{y}|\mathcal{D}, \mathbf{x}, \mathcal{X}^\star\right]$ también es la entropía de las distribuciones predictivas después de medir en $\mathbf{x}$ pero condicionada además a los valores de $\mathcal{X}^\star$, y ahora la esperanza $\mathbb{E}[\cdot]$ es respecto de posibles $\mathcal{X}^\star$ dado el conjunto de datos $\mathcal{D}$. Esta nueva ecuación (4.3) es más sencilla de aproximar que (4.2) porque no es necesario calcular la entropía de $\mathcal{X}^\star$. De hecho, (4.3) favorece evaluar en zonas del espacio de entradas $\mathcal{X}$ donde $\mathcal{X}^\star$ ofrece más información sobre $\mathbf{y}$, y estas regiones son también en las que es $\mathbf{y}$ da más información sobre $\mathcal{X}^\star$. La expresión (4.3) es justamente la función de adquisición aproximada por PESMOC.

Como se dijo anteriormente, el primer término del lado derecho de la ecuación (4.3), es simplemente la entropía de las distribuciones predictivas en el momento que se está decidiendo dónde evaluar

en la siguiente iteración, es decir:

$$\begin{aligned} H\left[p(\mathbf{y}|\mathcal{D},\mathbf{x})\right] &= \sum_{k=1}^{K}\frac{1}{2}\log(2\pi e v_k^f) + \sum_{j=1}^{C}\frac{1}{2}\log(2\pi e v_j^c) \\ &= \frac{K+C}{2}\log(2\pi e) + \frac{1}{2}\sum_{k=1}^{K}\log(v_k^f) + \frac{1}{2}\sum_{j=1}^{C}\log(v_j^c) \end{aligned} \tag{4.4}$$

donde v_k^f y v_j^c son las varianzas de las k-ésima y c-ésima distribuciones predictivas de las funciones objetivo y las restricciones, respectivamente. Se puede apreciar que para el cálculo de la entropía no son necesarias las medias m_k^f y m_j^c.

En contraposición con el primer término, el segundo término, de (4.3) no es sencillo de obtener debido a que es intratable, así que debe ser aproximado. Por un lado, la esperanza se puede aproximar mediante Montecarlo, es decir, generando muestras aleatorias de $\mathcal{X}^\star$, calculando la entropía de $p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{X}^\star)$ para esa muestra y promediando los resultados. Esas muestras de $\mathcal{X}^\star$, por ejemplo, se podrían obtener generando funciones de las distribuciones predictivas de los objetivos y las restricciones, y optimizando las soluciones mediante una rejilla de $D \times 1000$, donde D es la dimensión de los puntos de entrada. Esta optimización consiste en quedarse con las observaciones de la rejilla que no hayan estado Pareto-dominadas por ninguna otra observación y que sean válidas, y estos serán el conjunto de valores de la muestra de $\mathcal{X}^\star$. Este es el método que se ha utilizado en la implementación. Por otro lado, también se debe añadir, que en problemas donde la dimensión de entrada sea elevada, se puede utilizar un algoritmo evolutivo como NSGA-II [32] en lugar de una rejilla. Por otro lado, el cálculo de $p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{X}^\star)$ también debe ser aproximado, su aproximación se realizará en la siguiente sección.

4.3.1. Aproximación de la distribución condicionada al conjunto de Pareto solución

La distribución condicionada de los $\mathbf{y}$ al conjunto de datos $\mathcal{D}$, el punto donde se va a evaluar $\mathbf{x}$ y el conjunto de Pareto solución $\mathcal{X}^\star$, viene dada por la siguiente expresión:

$$p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{X}^\star) \propto \int p(\mathbf{y}|\mathbf{x},\mathbf{f},\mathbf{c})p(\mathcal{X}^\star|\mathbf{f},\mathbf{c})p(\mathbf{f}|\mathcal{D},\mathbf{x})p(\mathbf{c}|\mathcal{D},\mathbf{x})d\mathbf{f}d\mathbf{c} \tag{4.5}$$

donde $p(\mathbf{y}|\mathbf{x},\mathbf{f},\mathbf{c})$ es la probabilidad de observar el valor $\mathbf{y}$ dado el punto de entrada $\mathbf{x}$ y los verdaderos valores de los objetivos y las restricciones, $p(\mathcal{X}^\star|\mathbf{f},\mathbf{c})$ es la probabilidad de que $\mathcal{X}^\star$ sea el conjunto solución dado $\mathbf{f}$ y $\mathbf{c}$, y $p(\mathbf{f}|\mathcal{D},\mathbf{x})$ y $p(\mathbf{c}|\mathcal{D},\mathbf{x})$ son las probabilidades de haber obtenido $\mathbf{f}$ y $\mathbf{c}$, respectivamente, dado el conjunto de datos $\mathcal{D}$ y el punto donde se evalúa $\mathbf{x}$. Como se va a considerar un escenario donde no hay ruido, el valor de $\mathbf{y}$ es exactamente el de $(\mathbf{f},\mathbf{c})$, es decir:

$$p(\mathbf{y}|\mathbf{x},\mathbf{f},\mathbf{c}) = \prod_{k=1}^{K}\delta(y_k - f_k(\mathbf{x}))\prod_{j=1}^{C}\delta(y_{K+j} - c_j(\mathbf{x})) \tag{4.6}$$

donde $\delta(\cdot)$ es la función delta de Dirac. Para el caso donde hay ruido, simplemente habría que sustituir las deltas por Gaussianas.

El factor $p(\mathcal{X}^\star|\mathbf{f},\mathbf{c})$ se puede interpretar como la probabilidad de que cualquier punto $\mathbf{x}^\star$ pertenezca al conjunto solución $\mathcal{X}^\star$ dados los valores de las funciones objetivo y las restricciones. Por lo tanto, debe ser 0 para todos los conjuntos $\mathcal{X}^\star$ que no cumplan las restricciones, y para aquellos que aunque las cumplan, todos sus valores en las funciones objetivo sean superados por los valores obtenidos en otro punto. Es decir, solo puede valer 1 para los puntos que efectivamente pertenezcan a $\mathcal{X}^\star$, porque no están Pareto-dominados por ningún otro, y además cumplan las restricciones. Esto se puede expresar informalmente de la siguiente forma:

$$p(\mathcal{X}^\star|\mathbf{f},\mathbf{c}) = \prod_{\mathbf{x}^\star\in\mathcal{X}^\star}\left(\left(\prod_{j=1}^{C}\Phi_j\left(\mathbf{x}^\star\right)\right)\left(\prod_{\mathbf{x}'\in\mathcal{X}}\Omega\left(\mathbf{x}',\mathbf{x}^\star\right)\right)\right) \tag{4.7}$$

donde $\Phi_j(\cdot) = \Theta(c_j(\cdot))$, siendo $\Theta(\cdot)$ la función Heaviside o escalón (utilizando la convención $\Theta(0) = 1$), y la función $\Omega(\cdot,\cdot)$ viene dada por la siguiente ecuación:

$$\begin{aligned}\Omega\left(\mathbf{x}',\mathbf{x}^\star\right) &= \left(\prod_{j=1}^{C}\Phi_j\left(\mathbf{x}'\right)\right)\Psi\left(\mathbf{x}',\mathbf{x}^\star\right) + \left(1-\prod_{j=1}^{C}\Phi_j\left(\mathbf{x}'\right)\right)\cdot 1,\\ \Psi\left(\mathbf{x}',\mathbf{x}^\star\right) &= 1-\prod_{k=1}^{K}\Theta\left(f_k(\mathbf{x}^\star)-f_k(\mathbf{x}')\right).\end{aligned} \tag{4.8}$$

A continuación se dará una breve explicación de (4.7). El término $\prod_{j=1}^{C}\Phi_j(\mathbf{x}^\star)$ se ocupa de comprobar que todos los puntos $\mathbf{x}^\star$ son válidos, en caso de que no lo sean $p(\mathcal{X}^\star|\mathbf{f},\mathbf{c})$ valdrá 0, pero si lo son se continua con las comprobaciones. El término $\prod_{j=1}^{C}\Phi_j\left(\mathbf{x}'\right)$ (de (4.8)) se encarga de verificar que cualquier punto $\mathbf{x}'$, que se vaya a comparar con $\mathbf{x}^\star$, sea válido. Si no lo es, entonces no importa cuales sean los valores de las funciones objetivo en $\mathbf{x}'$ porque $\mathbf{x}^\star$ será mejor, por ser válido. Así que, simplemente, se devuelve el valor de $(1-\prod_{j=1}^{C}\Phi_j(\mathbf{x}'))$, que se ha multiplicado por 1 para denotar que devuelve 1. Por último, el término $\Psi\left(\mathbf{x}',\mathbf{x}^\star\right)$ confirma que los valores en $\mathbf{x}^\star$ efectivamente son superiores o iguales a los de $\mathbf{x}'$ en al menos una de las funciones objetivo (así que los dominan), en cuyo caso $p(\mathcal{X}^\star|\mathbf{f},\mathbf{c})$ valdrá 1.

Sustituyendo (4.6) y (4.7) en (4.5) se obtiene:

$$\begin{aligned}p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{X}^\star) \propto \int &\prod_{k=1}^{K}\delta(y_k-f_k(\mathbf{x}))\prod_{j=1}^{C}\delta(y_{K+j}-c_j(\mathbf{x}))\\ &\times\prod_{\mathbf{x}^\star\in\mathcal{X}^\star}\prod_{j=1}^{C}\Phi_j\left((\mathbf{x}^\star)\right)\prod_{\mathbf{x}^\star\in\mathcal{X}^\star}\left(\Omega\left(\mathbf{x},\mathbf{x}^\star\right)\prod_{\mathbf{x}'\in\mathcal{X}\setminus\{\mathbf{x}\}}\Omega\left(\mathbf{x}',\mathbf{x}^\star\right)\right)\\ &\times p(\mathbf{f}|\mathcal{D},\mathbf{x})p(\mathbf{c}|\mathcal{D},\mathbf{x})d\mathbf{f}d\mathbf{c}\end{aligned} \tag{4.9}$$

donde se ha separado en dos partes los factores del productorio de $\Omega(\cdot,\cdot)$ que dependen de $\mathbf{x}$ y los que no.

El cálculo de (4.9) tiene dos dificultades. La primera es que el conjunto de puntos $\mathcal{X}$ tiene un tamaño infinito. Esto se puede solventar aproximando $\mathcal{X}$ como $\hat{\mathcal{X}} = \{\mathbf{x}_n\}_{n=1}^{N} \cup \mathcal{X}^\star \cup \mathbf{x}$, donde $\{\mathbf{x}_n\}_{n=1}^{N}$ son los puntos observados hasta el momento. La segunda es que, para realizar ese cálculo de forma analítica, todos los términos deben ser Gaussianas, sin embargo, solamente tres lo son ($p(\mathbf{y}|\mathbf{x}, \mathbf{f}, \mathbf{c})$, $p(\mathbf{f}|\mathcal{D}, \mathbf{x})$ y $p(\mathbf{c}|\mathcal{D}, \mathbf{x})$). Por lo que hay que aproximar los factores $\Phi_j(\cdot)$ y $\Omega(\cdot, \cdot)$ del término $p(\mathcal{X}^\star|\mathbf{f}, \mathbf{c})$ con Gaussianas. Esto se realiza aplicando Expectation Propagation (EP) [33]. Al aplicar EP sobre el factor $\Phi_j(\cdot)$, este es aproximado como una Gaussiana unidimensional sin normalizar sobre $c_j(\mathbf{x}^\star)$:

$$\Phi_j(\mathbf{x}^\star) \approx \tilde{\Phi}_j(\mathbf{x}^\star) \propto \exp\left\{-\frac{1}{2} c_j(\mathbf{x}^\star)^2 \tilde{v}_j^{\mathbf{x}^\star} + c_j(\mathbf{x}^\star) \tilde{m}_j^{\mathbf{x}^\star}\right\}$$

donde $\tilde{v}_j^{\mathbf{x}^\star}$ y $\tilde{m}_j^{\mathbf{x}^\star}$ son los parámetros naturales de las Gaussianas ajustadas por EP. Por otra parte, al aplicar EP sobre $\Omega(\cdot, \cdot)$, este factor es aproximado como un producto de C Gaussianas unidimensionales y K Gaussianas bidimensionales:

$$\begin{aligned}\Omega(\mathbf{x}', \mathbf{x}^\star) \approx \tilde{\Omega}(\mathbf{x}', \mathbf{x}^\star) \propto & \prod_{k=1}^{K} \exp\left\{-\frac{1}{2} \boldsymbol{v}_k^{\mathrm{T}} \tilde{\mathbf{V}}_k^{\Omega} \boldsymbol{v}_k + \boldsymbol{v}_k \tilde{\mathbf{m}}_k^{\Omega}\right\} \\ & \times \prod_{k=1}^{K} \exp\left\{-\frac{1}{2} c_j(\mathbf{x}^\star)^2 \tilde{v}_j^{\Omega} + c_j(\mathbf{x}^\star) \tilde{m}_j^{\Omega}\right\}\end{aligned}$$

donde $\tilde{\mathbf{V}}_k^{\Omega}$ y $\tilde{\mathbf{m}}_k^{\Omega}$ son los parámetros naturales de una de las K Gaussianas bidimensionales ajustadas por EP, $\tilde{v}_j^{\Omega}$ y $\tilde{m}_j^{\Omega}$ son los parámetros naturales de una de las C Gaussianas unidimensionales también ajustadas por EP, y $\boldsymbol{v}_k = (f_k(\mathbf{x}'), f_k(\mathbf{x}^\star))^{\mathrm{T}}$.

El algoritmo de EP se encarga de refinar iterativamente todos esos parámetros naturales de las Gaussianas con el objetivo de que sean lo más parecidos posibles a los parámetros reales. Para acelerar EP, y que así no tarde demasiado la ejecución de PESMOC, todos los parámetros de los factores que no dependen de $\mathbf{x}$ son calculados una única vez y reutilizados en cada iteración para nuevos $\mathbf{x}$, y los que sí dependen de $\mathbf{x}$ solamente son actualizados una vez.

4.3.2. La función de adquisición PESMOC

Cuando el algoritmo de EP termina se obtienen los parámetros de las Gaussianas que aproximaban el término (4.7). Así que, ya se puede multiplicar este término aproximado por Gaussianas con el resto de términos de (4.9) que ya eran Gaussianas ($p(\mathbf{f}|\mathcal{D}, \mathbf{x})$, $p(\mathbf{c}|\mathcal{D}, \mathbf{x})$ y las deltas). Como la Gaussiana pertenece a la familia exponencial, el resultado de esas multiplicaciones es otra distribución Gaussiana, la cual viene definida por:

$$p(\mathbf{y}|\mathcal{D}, \mathbf{x}, \mathcal{X}^\star) \approx \prod_{k=1}^{K} \mathcal{N}(f_k(\mathbf{x})|\tilde{m}_k^f, \tilde{v}_k^f) \prod_{j=1}^{C} \mathcal{N}(c_j(\mathbf{x})|\tilde{m}_j^c, \tilde{v}_j^c) \tag{4.10}$$

donde $\tilde{m}_k^f$, $\tilde{v}_k^f$, $\tilde{m}_j^c$ y $\tilde{v}_j^c$ son los parámetros de las Gaussianas resultantes.

Ahora, ya es posible escribir la función de adquisición final de PESMOC, la cual simplemente es la diferencia entre las entropías antes y después de condicionar a una muestra de $\mathcal{X}^\star$.

$$\alpha(\mathbf{x}) \approx \sum_{k=1}^{K} \log(v_k^f(\mathbf{x})) + \sum_{j=1}^{C} \log(v_j^c(\mathbf{x})) - \frac{1}{M}\sum_{m=1}^{M}\left[\sum_{k=1}^{K} \log(\tilde{v}_k^f(\mathbf{x}|\mathcal{X}_{(m)}^\star)) + \sum_{j=1}^{C} \log(\tilde{v}_j^c(\mathbf{x}|\mathcal{X}_{(m)}^\star))\right]$$

donde $\{\mathcal{X}_{(m)}^\star\}_{m=1}^M$ son las M muestras Montecarlo generadas de $\mathcal{X}^\star$, para aproximar la esperanza de (4.3). A fin de tener en cuenta el ruido, simplemente se deben sumar las varianzas de los objetivos y las restricciones $(\boldsymbol{\sigma}^f)^2$ y $(\boldsymbol{\sigma}^c)^2$, respectivamente, a las varianzas $\tilde{\mathbf{v}}^f$ y $\tilde{\mathbf{v}}^c$ de las distribuciones predictivas condicionadas a $\{\mathcal{X}_{(m)}^\star\}_{m=1}^M$, es decir:

$$\begin{aligned}\alpha(\mathbf{x}) \approx & \sum_{k=1}^{K} \log(v_k^f(\mathbf{x}) + (\sigma_k^f)^2) + \sum_{j=1}^{C} \log(v_j^c(\mathbf{x}) + (\sigma_j^c)^2) \\ & - \frac{1}{M}\sum_{m=1}^{M}\left[\sum_{k=1}^{K} \log(\tilde{v}_k^f(\mathbf{x}|\mathcal{X}_{(m)}^\star) + (\sigma_k^f)^2) + \sum_{j=1}^{C} \log(\tilde{v}_j^c(\mathbf{x}|\mathcal{X}_{(m)}^\star) + (\sigma_j^c)^2)\right]\end{aligned} \tag{4.11}$$

Como la función adquisición está formulada como una suma de las funciones de adquisición de los objetivos y las restricciones, se pueden comparar estas funciones, para elegir en qué f o c evaluar, además de en qué punto. Esto permite utilizar PESMOC en entornos desacoplados. El objetivo o restricción donde se decida evaluar, será aquel cuyo máximo sea el mayor de entre todos los máximos. Como este enfoque permite que evaluar exactamente en el punto que se espera reduzca más la entropía de la solución de cada objetivo y restricción, en lugar de en el punto cuya suma de funciones de adquisición es mayor (punto que no tiene por qué ser el máximo de ninguna función), se espera que esta opción obtenga unos resultados superiores a los del enfoque acoplado.

4.4. Búsqueda de la entropía del máximo valor con múltiples objetivos y restricciones desconocidas

En esta sección se realizará el desarrollo de MESMOC, la función de adquisición presentada en este trabajo. Esta función, puede trabajar en problemas con múltiples objetivos sujetos a varias restricciones al igual que PESMOC. Sin embargo, la derivación de MESMOC no es tan compleja, y además, no cuenta con las aproximaciones que se realizan mediante EP, así que es menos costosa y más sencilla de implementar.

Al igual que en el desarrollo de PESMOC, la derivación de MESMOC, también se llevará a cabo suponiendo la versión acoplada del método, y al final de la Sección 4.4.2 se tratará su funcionamiento de forma desacoplada. Además, se supondrá que no hay ruido, es decir, que se observan los verdaderos valores de $\mathbf{f}$ y $\mathbf{c}$.

Al igual que PESMOC, MESMOC también se basa en la reducción de la entropía de la solución. Sin embargo, en lugar de utilizar las localizaciones de los puntos no Pareto-dominados por ningún otro, usa los valores de esos puntos, es decir, en lugar de usar el conjunto de Pareto solución $\mathcal{X}^\star$, emplea la frontera de Pareto solución $\mathcal{Y}^\star$. Al realizar este cambio en la función de adquisición PESMOC, expresada en la ecuación (4.3), se obtiene la función de adquisición MESMOC, en la cual la información mutua es entre $\mathcal{Y}^\star$ e $\mathbf{y}$, y su expresión es:

$$\alpha(\mathbf{x}) = H\left[p(\mathbf{y}|\mathcal{D},\mathbf{x})\right] - \mathbb{E}_{p(\mathcal{Y}^\star|\mathcal{D})}\left[H\left[p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{Y}^\star)\right]\right] \tag{4.12}$$

donde $p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{Y}^\star)$ es la probabilidad de $\mathbf{y}$ dado el conjunto de datos $\mathcal{D}$, el punto donde se evalúa $\mathbf{x}$ y la frontera de Pareto $\mathcal{Y}^\star$, y la esperanza es respecto de la frontera de Pareto $\mathcal{Y}^\star$ dado el conjunto de datos $\mathcal{D}$. Se puede apreciar que el primer término del lado derecho de (4.12) es exactamente el mismo que el de (4.3). Por lo que, al igual que antes, este es simplemente la entropía de las distribuciones predictivas, y su expresión es (4.4). Sin embargo, tal como ocurrió con PESMOC, el segundo término del lado derecho de (4.12) es intratable. La esperanza se puede aproximar generando muestras Montecarlo de $\mathcal{Y}^\star$, calculando la entropía de $p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{Y}^\star)$ y promediando los resultados. La generación de muestras de $\mathcal{Y}^\star$ se realiza optimizando los valores obtenidos de una rejilla de $D \times 1000$, donde D es la dimensión de entrada. La optimización consiste en quedarse con los valores $\mathbf{y}^\star$ de los puntos no dominados $\mathbf{x}^\star$ que sean válidos. Por otra parte, el cálculo de $p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{Y}^\star)$ debe ser aproximado. Esta aproximación se efectúa en la siguiente sección.

4.4.1. Aproximación de la distribución condicionada a la frontera de Pareto solución

De la misma forma que en el caso de MES, se considerará que las observaciones no están contaminadas por ruido en la derivación, aunque en realidad sí que lo estén, es decir, se aproximará $p(\mathbf{f},\mathbf{c}|\mathcal{D},\mathbf{x},\mathcal{Y}^\star)$ en lugar de $p(\mathbf{y}|\mathcal{D},\mathbf{x},\mathcal{Y}^\star)$. El ruido de las observaciones se añadirá al final de de la Sección 4.4.2 tal como se hace en el desarrollo de PESMOC de la Sección 4.3.2. La expresión de $p(\mathbf{f},\mathbf{c}|\mathcal{D},\mathbf{x},\mathcal{Y}^\star)$ viene dada por:

$$p(\mathbf{f},\mathbf{c}|\mathcal{D},\mathbf{x},\mathcal{Y}^\star) = Z^{-1}p(\mathbf{f},\mathbf{c}|\mathcal{D},\mathbf{x})p(\mathcal{Y}^\star|\mathbf{f},\mathbf{c}) \tag{4.13}$$

donde Z^{-1} es una constante de normalización, $p(\mathbf{f},\mathbf{c}|\mathcal{D},\mathbf{x})$ es la probabilidad de los objetivos y restricciones dado el conjunto de datos y el punto donde se evalúa, y $p(\mathcal{Y}^\star|\mathbf{f},\mathbf{c})$ es la probabilidad de la frontera de Pareto dados $\mathbf{f}$ y $\mathbf{c}$.

El factor $p(\mathcal{Y}^\star|\mathbf{f},\mathbf{c})$ de (4.13) se encarga de eliminar todas las configuraciones de $\mathbf{f}$ y $\mathbf{c}$ que son incompatibles con la frontera de Pareto óptima $\mathcal{Y}^\star$. El valor de $p(\mathcal{Y}^\star|\mathbf{f},\mathbf{c})$ debe ser 0 cuando $\mathbf{f}$ o $\mathbf{c}$ sean inválidos, es decir, cuando no se cumpla que $c_j(\mathbf{x}) \geq 0, \forall j \in \{1,...,C\}$, o los valores de $\mathbf{f}$ sean

menores o iguales en todos los objetivos que los de algún punto de la frontera, excepto un valor que debe ser estrictamente menor. De forma equivalente, el factor $p(\mathcal{Y}^\star|\mathbf{f},\mathbf{c})$ valdrá 1 cuando todos los $\mathbf{f}^\star$ del conjunto $\mathcal{Y}^\star$ sean menores en al menos un objetivo que $\mathbf{f}$, y los valores de $\mathbf{c}$ sean todos mayores a 0. Esto se puede expresar informalmente de la siguiente manera:

$$\begin{aligned} p(\mathcal{Y}^\star|\mathbf{f},\mathbf{c}) &= \prod_{\mathbf{f}^\star\in\mathcal{Y}^\star}\left(1-\prod_{j=0}^{C}\Theta(c_j(\mathbf{x}))\prod_{k=0}^{K}\Theta\left(f_k^\star-f_k(\mathbf{x})\right)\right) \\ &= \prod_{\mathbf{f}^\star\in\mathcal{Y}^\star}\Omega(\mathbf{f}^\star,\mathbf{f},\mathbf{c}) \end{aligned} \tag{4.14}$$

donde $\Omega(\mathbf{f}^\star,\mathbf{f},\mathbf{c}) = 1-\prod_{j=0}^{C}\Theta(c_j(\mathbf{x}))\prod_{k=0}^{K}\Theta\left(f_k^\star-f_k(\mathbf{x})\right)$, y $f_k^\star$ es el mejor valor de la k-ésima función objetivo. Se puede ver que (4.14) solamente valdrá 1, si el valor de $\Omega(\mathbf{f}^\star,\mathbf{f},\mathbf{c})$ para todas las $\mathbf{f}^\star$ es 1. Asimismo, $\Omega(\mathbf{f}^\star,\mathbf{f},\mathbf{c})$ vale 1 cuando $\prod_{j=0}^{C}\Theta(c_j(\mathbf{x}))$ y $\prod_{k=0}^{K}\Theta\left(f_k^\star-f_k(\mathbf{x})\right)$ valen 0, y eso sucede si todos los valores de $\mathbf{c}$ son mayores o iguales a 0 y todos los valores de $\mathbf{f}^\star$ son menores o iguales a los de $\mathbf{f}$, excepto uno que debe ser estrictamente menor.

No se puede calcular (4.13) de forma analítica, dado que el cálculo de $p(\mathcal{Y}^\star|\mathbf{f},\mathbf{c})$ implica varias funciones escalón que dependen del mismo $\mathbf{f}^\star$, por consiguiente, se debe aproximar (4.13). Un enfoque para llevar a cabo esta aproximación consiste en tener en cuenta solamente el mejor valor de cada objetivo, tal como se hace en MESMO [19]. Sin embargo, se ha visto que este enfoque da malos resultados porque es bastante restrictivo. En su lugar, la aproximación de (4.13) se ha realizado utilizando Gaussianas truncadas obtenidas mediante Assumed Density Filtering (ADF) [34].

ADF es una versión de EP que solo procesa los factores una vez [33]. Al usar ADF se quiere aproximar una distribución $\hat{p}(\mathbf{a})$ utilizando una $q(\mathbf{a})$. La distribución $q(\mathbf{a})$ ya pertenece a la familia de distribuciones deseada (en este caso es una Gaussiana). Para conseguir que $q(\mathbf{a})$ aproxime a $\hat{p}(\mathbf{a})$, el algoritmo de ADF minimiza la divergencia de Kullback-Leibler entre $\hat{p}(\mathbf{a})$ y $q(\mathbf{a})$, es decir, se minimiza $KL(\hat{p}(\mathbf{a})||q(\mathbf{a}))$. Al estar trabajando con una distribución que pertenece a la familia exponencial, esta minimización consiste en hacer coincidir momentos (matching moments) entre las dos distribuciones. Es decir, se va a utilizar el algoritmo de ADF para alterar las medias y las varianzas de las distribuciones predictivas para que estas distribuciones se parezcan a unas que están condicionadas a la frontera de Pareto $\mathcal{Y}^\star$. Esta modificación de las medias y varianzas se realiza procesando iterativamente todos los puntos de $\mathcal{Y}^\star$. Por otra parte, como (4.13) se puede expresar de la siguiente manera:

$$Z = \int t(\mathbf{a})\mathcal{N}(\mathbf{a}|\boldsymbol{\mu},\boldsymbol{\Sigma})d\mathbf{a}, \qquad \hat{p}(\mathbf{a}) = Z^{-1}t(\mathbf{a})\mathcal{N}(\mathbf{a}|\boldsymbol{\mu},\boldsymbol{\Sigma})$$

dado que $\Omega(\mathbf{f}^\star,\mathbf{f},\mathbf{c})$ es $t(\mathbf{a})$ y las distribuciones predictivas de los procesos Gaussianos son $\mathcal{N}(\mathbf{a}|\boldsymbol{\mu},\boldsymbol{\Sigma})$, donde $\boldsymbol{\mu}$ y $\boldsymbol{\Sigma}$ son el vector de medias y la matriz de covarianzas de la Gaussiana, entonces, se pueden utilizar las ecuaciones del Apéndice 5 de [35] para actualizar iterativamente $q(\mathbf{a})$:

$$\begin{aligned}\mathbb{E}_{\hat{p}(\mathbf{a})}[\mathbf{a}] &= \boldsymbol{\mu} + \boldsymbol{\Sigma}\frac{\partial \log(Z)}{\partial \boldsymbol{\mu}} \\ \mathbb{E}_{\hat{p}(\mathbf{a})}[\mathbf{a}\mathbf{a}^{\mathrm{T}}] - \mathbb{E}_{\hat{p}(\mathbf{a})}[\mathbf{a}]\mathbb{E}_{\hat{p}(\mathbf{a})}[\mathbf{a}]^{\mathrm{T}} &= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\left(\frac{\partial \log(Z)}{\partial \boldsymbol{\mu}}\left(\frac{\partial \log(Z)}{\partial \boldsymbol{\mu}}\right)^{\mathrm{T}} - 2\frac{\partial \log(Z)}{\partial \boldsymbol{\Sigma}}\right)\boldsymbol{\Sigma}.\end{aligned} \tag{4.15}$$

donde Z es una constante de normalización.

Por lo tanto, para utilizar ADF, se debe calcular Z y las derivadas parciales de $\log(Z)$ respecto de las medias y las varianzas de las funciones objetivo y restricciones. El cálculo de Z es el siguiente:

$$\begin{aligned}Z &= \int p(\mathbf{f}, \mathbf{c}|\mathcal{D}, \mathbf{x})\Omega(\mathbf{f}^{\star}, \mathbf{f}, \mathbf{c})d\mathbf{f}d\mathbf{c} \\ &= \int p(\mathbf{f}, \mathbf{c}|\mathcal{D}, \mathbf{x})\left(1 - \prod_{j=0}^{C}\Theta(c_j(\mathbf{x}))\prod_{k=0}^{K}\Theta\left(f_k^{\star} - f_k(\mathbf{x})\right)\right)d\mathbf{f}d\mathbf{c} \\ &= 1 - \prod_{j=0}^{C}\Phi(\gamma_j^c)\prod_{k=0}^{K}\Phi(\gamma_k^f)\end{aligned} \tag{4.16}$$

donde $\Phi(\cdot)$ es la distribución de probabilidad acumulada de la Gaussiana, y

$$\gamma_k^f = \frac{f_k^{\star} - m_k^f(\mathbf{x})}{(v_k^f(\mathbf{x}))^{1/2}} \qquad\qquad \gamma_j^c = \frac{m_j^c(\mathbf{x})}{(v_j^c(\mathbf{x}))^{1/2}}$$

donde $m_k^f(\mathbf{x})$, $v_k^f(\mathbf{x})$, $m_k^c(\mathbf{x})$ y $v_k^c(\mathbf{x})$ son los valores de la media y la varianza de las k-ésima y j-ésima distribuciones predictivas de los objetivos y las restricciones en el punto $\mathbf{x}$. Por otro lado, estos son los valores de las derivadas $\frac{\partial \log(Z)}{\partial m_k^f}$, $\frac{\partial \log(Z)}{\partial v_k^f}$, $\frac{\partial \log(Z)}{\partial m_j^c}$ y $\frac{\partial \log(Z)}{\partial v_j^c}$:

$$\frac{\partial \log(Z)}{\partial m_k^f} = \frac{(Z-1)}{Z\Phi(\gamma_k^f)}\left(\frac{-\mathcal{N}(\gamma_k^f|0,1)}{(v_k^f)^{1/2}}\right) \qquad \frac{\partial \log(Z)}{\partial v_k^f} = \frac{(Z-1)}{Z\Phi(\gamma_k^f)}\left(\mathcal{N}(\gamma_k^f|0,1)\frac{-\gamma_k^f}{2v_k^f}\right) \tag{4.17}$$

$$\frac{\partial \log(Z)}{\partial m_j^c} = \frac{(Z-1)}{Z\Phi(\gamma_j^c)}\left(\frac{\mathcal{N}(\gamma_j^c|0,1)}{(v_j^c)^{1/2}}\right) \qquad \frac{\partial \log(Z)}{\partial v_j^c} = \frac{(Z-1)}{Z\Phi(\gamma_j^c)}\left(\mathcal{N}(\gamma_j^c|0,1)\frac{-\gamma_j^c}{2v_j^c}\right) \tag{4.18}$$

donde $v_k^f = v_k^f(\mathbf{x})$ y $v_j^c = v_j^c(\mathbf{x})$. Su desarrollo completo está en el Apéndice A.

El algoritmo de ADF se muestra en el Algoritmo 4.1. Se puede ver que en cada iteración se actualizan los valores de $\tilde{\mathbf{m}}^f$, $\tilde{\mathbf{v}}^f$, $\tilde{\mathbf{m}}^c$ y $\tilde{\mathbf{v}}^c$ mediante las derivadas calculadas de las ecuaciones (4.17) y (4.18). También es posible apreciar que el orden en el que se procesen los $\mathbf{f}^{\star}$, influirá en el resultado de las medias y las varianzas. Por este motivo, se ha establecido este orden como aleatorio.

Por otra parte, una vez termina el algoritmo de ADF, ya se han actualizado todas las medias y varianzas, de las distribuciones predictivas, entonces, ya es posible obtener la expresión final de función de adquisición MESMOC, la cual será mostrada en la siguiente sección.

input : $\mathbf{m}^f$, $\mathbf{v}^f$, $\mathbf{m}^c$ y $\mathbf{v}^c$

Se inicializa: $\tilde{\mathbf{m}}^f = \mathbf{m}^f$, $\tilde{\mathbf{v}}^f = \mathbf{v}^f$, $\tilde{\mathbf{m}}^c = \mathbf{m}^c$ y $\tilde{\mathbf{v}}^c = \mathbf{v}^c$;

for each $\mathbf{f}^\star$ **in** $\mathcal{Y}^\star$ **do**

$\boldsymbol{\gamma}^f = (\mathbf{f}^\star - \mathbf{m}^f)/\sqrt{\mathbf{v}^f}$;

$\boldsymbol{\gamma}^c = \mathbf{m}^c/\sqrt{\mathbf{v}^c}$;

$Z = 1 - \prod_{j=0}^{C} \Phi(\gamma_j^c) \prod_{k=0}^{K} \Phi(\gamma_k^f)$;

$\tilde{\mathbf{m}}^f = \tilde{\mathbf{m}}^f + \tilde{\mathbf{v}}^f \frac{\partial \log(Z)}{\partial \tilde{\mathbf{m}}^f}$;

$\tilde{\mathbf{v}}^f = \tilde{\mathbf{v}}^f - \tilde{\mathbf{v}}^f \left(\frac{\partial \log(Z)}{\partial \tilde{\mathbf{m}}^f} \left(\frac{\partial \log(Z)}{\partial \tilde{\mathbf{m}}^f} \right)^{\mathrm{T}} - 2 \frac{\partial \log(Z)}{\partial \tilde{\mathbf{v}}^f} \right) \tilde{\mathbf{v}}^f$;

$\tilde{\mathbf{m}}^c = \tilde{\mathbf{m}}^c + \tilde{\mathbf{v}}^c \frac{\partial \log(Z)}{\partial \tilde{\mathbf{m}}^f}$;

$\tilde{\mathbf{v}}^c = \tilde{\mathbf{v}}^c - \tilde{\mathbf{v}}^c \left(\frac{\partial \log(Z)}{\partial \tilde{\mathbf{m}}^c} \left(\frac{\partial \log(Z)}{\partial \tilde{\mathbf{m}}^c} \right)^{\mathrm{T}} - 2 \frac{\partial \log(Z)}{\partial \tilde{\mathbf{v}}^c} \right) \tilde{\mathbf{v}}^c$;

end

return $\tilde{\mathbf{m}}^f$, $\tilde{\mathbf{v}}^f$, $\tilde{\mathbf{m}}^c$ y $\tilde{\mathbf{v}}^c$

Algoritmo 4.1: Algoritmo ADF.

4.4.2. La función de adquisición MESMOC

Al igual que en el caso de PESMOC, la expresión de (4.13) después de ser procesada queda de la siguiente forma

$$p(\mathbf{f}, \mathbf{c}|\mathcal{D}, \mathbf{x}, \mathcal{Y}^\star) \approx \prod_{k=1}^{K} \mathcal{N}(f_k(\mathbf{x})|\tilde{m}_k^f, \tilde{v}_k^f) \prod_{j=1}^{C} \mathcal{N}(c_j(\mathbf{x})|\tilde{m}_j^c, \tilde{v}_j^c) \tag{4.19}$$

donde $\tilde{m}_k^f$, $\tilde{v}_k^f$, $\tilde{m}_j^c$ y $\tilde{v}_j^c$ son las medias y varianzas de las distribuciones predictivas aproximadas condicionadas a la frontera $\mathcal{Y}^\star$.

Con el propósito de escribir la expresión final de MESMOC, la cual es una aproximación de (4.12), simplemente se debe restar a la expresión (4.4) un promedio de M muestras Montecarlo de la entropía de (4.19), de forma que se obtiene:

$$\alpha(\mathbf{x}) \approx \sum_{k=1}^{K} \log(v_k^f(\mathbf{x})) + \sum_{j=1}^{C} \log(v_j^c(\mathbf{x})) - \frac{1}{M} \sum_{m=1}^{M} \left[\sum_{k=1}^{K} \log(\tilde{v}_k^f(\mathbf{x}|\mathcal{Y}_{(m)}^\star)) + \sum_{j=1}^{C} \log(\tilde{v}_j^c(\mathbf{x}|\mathcal{Y}_{(m)}^\star)) \right]$$

donde $\{\mathcal{Y}_{(m)}^\star\}_{m=1}^{M}$ es el conjunto de muestras generadas de $\mathcal{Y}^\star$. Al igual que en el caso de PESMOC, para añadir el ruido solamente es necesario sumarlo a las varianzas de las distribuciones predictivas condicionadas, de modo que se obtiene:

$$\begin{aligned} \alpha(\mathbf{x}) \approx & \sum_{k=1}^{K} \log\left(v_k^f(\mathbf{x}) + (\sigma_k^f)^2\right) + \sum_{j=1}^{C} \log\left(v_j^c(\mathbf{x}) + (\sigma_j^c)^2\right) \\ & - \frac{1}{M} \sum_{m=1}^{M} \left[\left(\sum_{k=1}^{K} \log(\tilde{v}_k^f(\mathbf{x}|\mathcal{Y}_{(m)}^\star)) + (\sigma_k^f)^2 \right) + \left(\sum_{j=1}^{C} \log(\tilde{v}_j^c(\mathbf{x}|\mathcal{Y}_{(m)}^\star)) + (\sigma_j^c)^2 \right) \right]. \end{aligned}$$

Sin embargo, al utilizar esta expresión el comportamiento no era el esperado. Esto se debía a que MESMOC evaluaba muchas veces en los mismos puntos. Este problema estaba originado por la

reducción en la varianza que ocurre al procesar los puntos mediante ADF. Si un punto de la distribución predictiva tenía una varianza de 10^{-5}, su valor al aplicar el logaritmo sería $-11{,}51$. Asimismo, al aplicar ADF para obtener la distribución predictiva condicionada, ese punto podría pasar a tener una varianza de 10^{-6}, cuyo valor al aplicar el logaritmo es $-13{,}82$. Esto suponía una reducción muy alta en un punto donde desde el principio no hacia falta evaluar. A fin de evadir este tipo de situaciones, se decidió tener en cuenta la reducción absoluta de la varianza en lugar de su logaritmo. De forma que, la expresión final de MESMOC queda de la siguiente manera:

$$\begin{aligned}\alpha(\mathbf{x}) \approx & \sum_{k=1}^{K}\left(v_k^f(\mathbf{x}) + (\sigma_k^f)^2\right) + \sum_{j=1}^{C}\left(v_j^c(\mathbf{x}) + (\sigma_j^c)^2\right) \\ & - \frac{1}{M}\sum_{m=1}^{M}\left[\left(\sum_{k=1}^{K}\tilde{v}_k^f(\mathbf{x}|\mathcal{Y}_{(m)}^\star) + (\sigma_k^f)^2\right) + \left(\sum_{j=1}^{C}\tilde{v}_j^c(\mathbf{x}|\mathcal{Y}_{(m)}^\star) + (\sigma_j^c)^2\right)\right]\end{aligned} \tag{4.20}$$

Al igual que PESMOC, MESMOC está definida como la suma de la función de adquisición de cada objetivo y restricción por separado. Por lo tanto, también se puede usar de forma desacoplada, dado que es posible elegir no solo en qué lugar evaluar, sino también, en qué función o restricción. Esto permite enfocarse en las regiones de cada objetivo o restricción que ayuden más a reducir la entropía de la solución.

En la Figura 4.4 se muestra un ejemplo de ejecución de $\mathrm{MESMOC}_{\mathrm{des}}$ (la versión desacoplada de MESMOC), donde en cada fila hay un objetivo o restricción. Para elaborar esta figura se creó un problema sintético de una dimensión de entrada con dos objetivos y una restricción. Posteriormente, se realizaron 5 evaluaciones. Por lo tanto, en 4.4 se presenta la evaluación que realiza $\mathrm{MESMOC}_{\mathrm{des}}$ a fin de decidir dónde realizar la sexta observación. En la primera columna se muestra el estado actual de las distribuciones predictivas de los dos objetivos f_1 y f_2 y la restricción c_1. En la segunda columna también se presentan las distribuciones predictivas, además de 20 valores de una muestra de la frontera de Pareto $\mathcal{Y}^\star$ obtenida al optimizar los posteriores de los $\mathcal{GP}$s. En la tercera columna se presenta una aproximación mediante ADF de las distribuciones predictivas de f_1, f_2 y c_1 condicionadas a la muestra de $\mathcal{Y}^\star$ que aparece en la segunda columna. En la cuarta columna se presentan las funciones de adquisición de f_1, f_2 y c_1 por separado. Se puede observar que para cada objetivo y restricción existe un máximo distinto donde evaluar. Por otra parte, en la Figura 4.5 se presenta la función de adquisición total, obtenida al sumar las funciones de adquisición que aparecen en la tercera columna de la Figura 4.4. Se puede apreciar que el máximo de esta función de adquisición se sitúa en un punto diferente al máximo de las funciones de adquisición de cada caja negra.

El coste computacional de MESMOC es $\mathcal{O}(M(K+C)|\mathcal{Y}_{(m)}^\star|)$, donde M es el número de muestras Montecarlo, y K y C son el número de objetivos y restricciones respectivamente. La parte del coste correspondiente a $(K+C)|\mathcal{Y}_{(m)}^\star|$ viene de ejecutar ADF para aproximar las varianzas de las distribuciones predictivas condicionadas a $\mathcal{Y}^\star$. Dicha aproximación solamente se realiza una vez por

iteración.

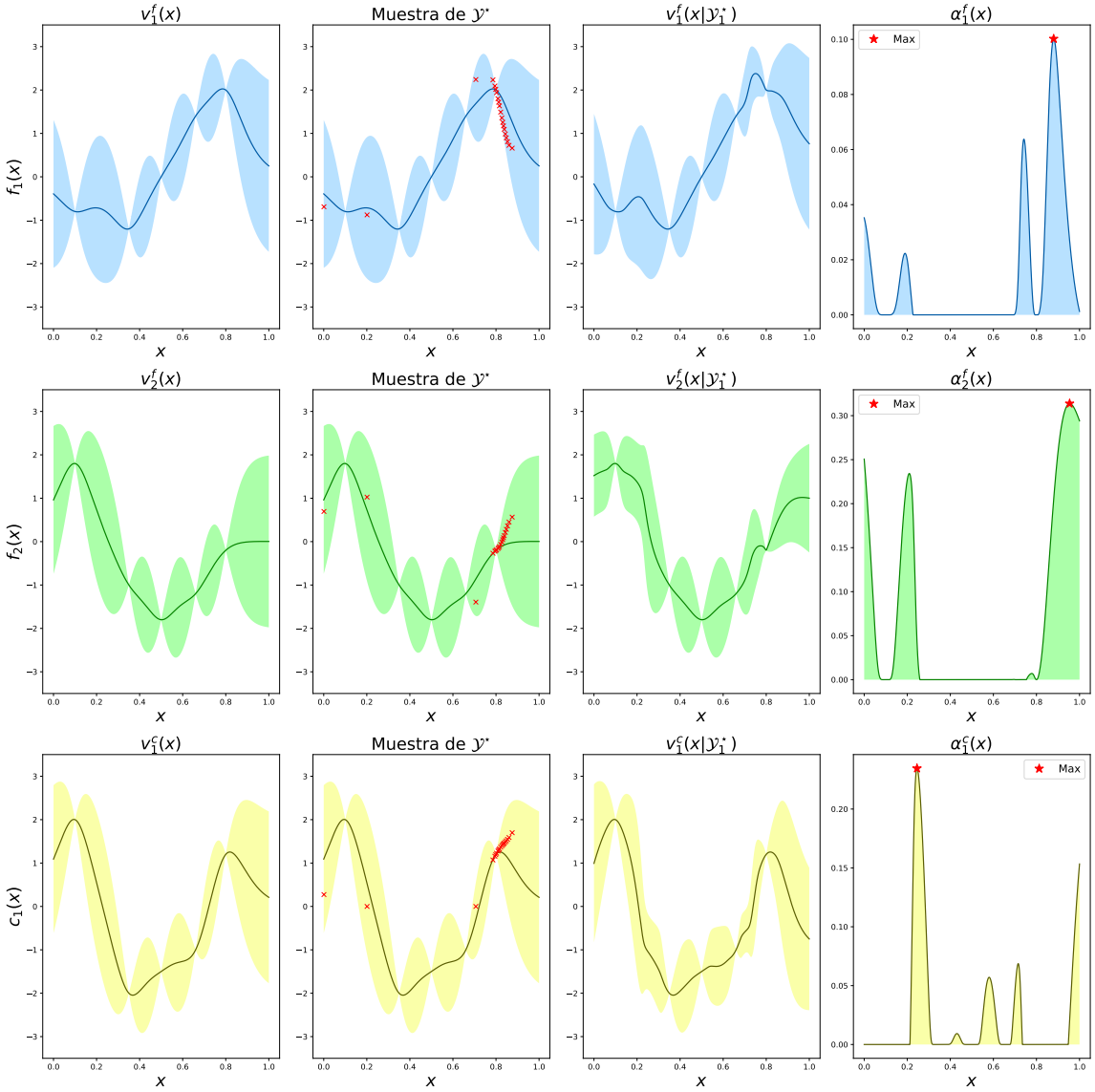


Figura 4.4: En la primera columna se muestran las distribuciones predictivas de los objetivos f_1 y f_2 y la restricción c_1. En la segunda columna además de las distribuciones predictivas, se muestran 20 puntos de una frontera de Pareto $\mathcal{Y}^\star$ obtenida tras optimizar los posteriores $\mathcal{GP}$s. En la tercera columna se presentan las distribuciones predictivas (de la primera columna) condicionadas a la muestra $\mathcal{Y}^\star$ de la segunda columna. En la cuarta columna aparece la utilidad esperada de evaluar en cada punto de entrada $\mathbf{x}$ para los dos objetivos y la restricción.

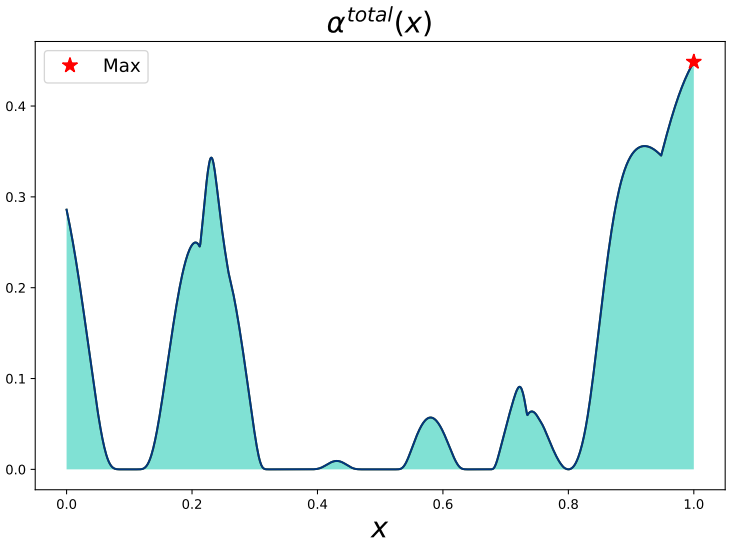


Figura 4.5: Se muestra la función de adquisición total del problema de la Figura 4.4. Se puede observar que corresponde exactamente a la suma de las funciones de adquisición de la cuarta columna de la Figura 4.4.

5

Experimentos realizados

Con el propósito de ver el desempeño de MESMOC, ha comparado su actuación con otros tres métodos en cuatro experimentos, dos generados sintéticamente y dos reales. Todos los experimentos realizados son los que aparecen en el artículo [5]. Además, las consideraciones prácticas que se describen (número de muestras generadas, validez de recomendaciones, etc.), y las estrategias con las que se compara son las que aparecen en dicho artículo.

Las funciones de adquisición con las que se ha comparado MESMOC, son PESMOC [5], BMOO [4] y RANDOM (una estrategia que elige aleatoriamente dónde evaluar). Se espera que RANDOM obtenga los peores resultados debido a que no utiliza el modelo de las funciones objetivo ni de las restricciones para guiar el proceso de optimización. Además, esta función de adquisición es equivalente a una exploración uniforme del espacio de parámetros. Por otro lado, la implementación de MESMOC ha sido realizada en el software de optimización Bayesiana Spearmint (https://github.com/HIPS/Spearmint) donde ya estaban implementadas las otras funciones de adquisición. Por otra parte, los resultados de los métodos en los diferentes experimentos realizados corresponden al promedio de 100 simulaciones diferentes para cada uno.

El kernel de los $\mathcal{GP}$s de todos los métodos es Matérn52, el cual se comenta en la Sección 2.2. Para obtener los hiperparámetros del kernel se utiliza slice sampling [25], un método que se comenta en la Sección 2.3, en el cual se ha establecido a 10 el número de muestras Montecarlo de la distribución posterior de los hiperparámetros. También se ha especificado como 10 el número de muestras Montecarlo del segundo término de la expresión de MESMOC, la cual se muestra en la ecuación (4.20). El tamaño de las muestras de los conjuntos generados de la frontera de Pareto $\mathcal{Y}^\star$ para MESMOC y el conjunto de Pareto $\mathcal{X}^\star$ para MESMOC y BMOO es 50. Por otro lado, para maximizar la función de adquisición se utiliza el método de optimización local L-BFGS, y las localizaciones de partida para este algoritmo han sido los puntos de una rejilla de $D \times 1000$, con D la dimensión de entrada. El cálculo de los gradientes de la función de adquisición que usa L-BFGS se ha aproximado por diferencias.

Para evitar que cualquiera de las funciones de adquisición recomienden puntos inválidos, solo se consideran aquellos $\mathbf{x}$ que cumplen la siguiente condición: $p(c_j(\mathbf{x})) \geq 1 - \delta, \forall j \in 1, ..., C$, donde δ es una constante que comienza valiendo $0{,}05$, y su valor es incrementado iterativamente en $0{,}05$ unidades

mientras que no haya ningún punto que cumpla dicha condición. Este mismo esquema es el que se utiliza en [5, 13, 16].

5.1. Experimentos con datos sintéticos

En este trabajo, se han llevado a cabo dos experimentos sintéticos a fin de ver el rendimiento de MESMOC. El primero de ellos ha contado con 2 funciones objetivo y 2 restricciones, y el espacio de entrada ha sido de 4 dimensiones. Mientras que el segundo ha tenido 4 objetivos y 2 restricciones, y el espacio de entrada ha sido de 6 dimensiones. En ambos experimentos las funciones objetivo y restricciones han sido generadas aleatoriamente mediante $\mathcal{GP}$s a priori, por lo que un modelo que utilice un $\mathcal{GP}$ se debería poder adaptar bien a estos objetivos y restricciones. En el primer experimento se realizan 400 evaluaciones y en el segundo 600. 400 evaluaciones equivalen a 100 iteraciones cuando se realiza una optimización de forma acoplada en 4 cajas negras, dado que en cada iteración se evalúa en todas cajas negras en el mismo punto. Sin embargo, 400 evaluaciones equivalen a 400 iteraciones cuando se realiza una optimización de forma desacoplada, porque en cada iteración solo se observa una caja negra en un solo punto, así que, son necesarias 400 iteraciones para realizar 400 evaluaciones. Por otra parte, cada experimento se realiza en un escenario donde las observaciones están contaminadas con ruido, y otro en el que no lo están. Este ruido es una variable aleatoria generada mediante una Gaussiana con desviación estándar $0{,}1$. Los resultados han sido obtenidos promediando 100 simulaciones distintas de cada método en cada experimento tanto para la versión con ruido como para la que no lo tiene, y las medias y desviaciones estándar que se muestran son estimadas utilizando 200 muestras por bootstrap. Al final, en estos experimentos se han generado 400 simulaciones distintas en total para cada método.

Después de cada iteración del algoritmo de BO, cada uno de los métodos comparados devuelve los puntos que conforman su conjunto de Pareto, en ese momento, como recomendación de la solución del problema. Este conjunto de Pareto se obtiene al optimizar las medias de las distribuciones predictivas a posteriori de los $\mathcal{GP}$. Posteriormente, para obtener los resultados, se evalúan los puntos de estos conjuntos en las funciones objetivo y restricciones reales para calcular el hipervolumen de la frontera de Pareto. El objetivo es maximizar este hipervolumen, porque tal como se dijo en la Sección 4.1 es una buena medida para ver del desempeño de diferentes estrategias en problemas con varios objetivos. En las iteraciones donde este conjunto contiene puntos inválidos se establece el hipervolumen a cero.

Los resultados del experimento 4d se muestran en la Figura 5.1. En ella se puede ver el promedio de la diferencia relativa del logaritmo respecto del hipervolumen de cada método en cada iteración con el hipervolumen de la solución real durante de 100 iteraciones. Los valores que se presentan son el promedio de los obtenidos en las 100 simulaciones generadas aleatoriamente para cada experimento. Se puede apreciar que los mejores resultados son obtenidos por MESMOC, PESMOC y las versiones desacopladas de cada uno $\mathrm{MESMOC}_{\mathrm{des}}$ y $\mathrm{PESMOC}_{\mathrm{des}}$, respectivamente, tanto en el entorno con ruido

como sin él. Por otro lado, BMOO obtiene unos resultados similares a los de las estrategias basadas en minimización de la entropía en el entorno sin ruido, sin embargo, cuando los valores observados están contaminados por ruido, su rendimiento empeora respecto de estos métodos y se asemeja al de RANDOM. Por otra parte, la búsqueda aleatoria obtiene los peores resultados tanto con ruido como sin él.

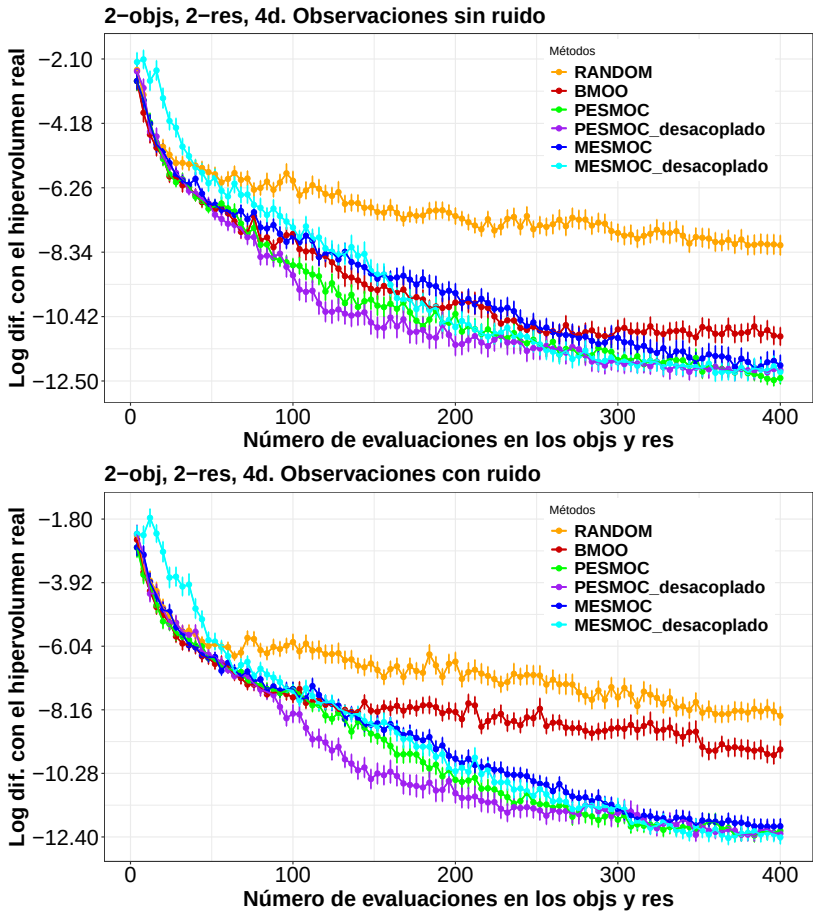


Figura 5.1: Se muestran los resultados del experimento sintético 4d. Los valores corresponden al promedio de 100 simulaciones de la diferencia en logaritmo del hipervolumen entre los conjuntos de cada método en cada iteración con el conjunto solución. Los conjuntos obtenidos por cada método en cada iteración se obtienen optimizando las medias de los $\mathcal{GP}$s a posteriori.

En la Figura 5.2, se presentan las medias y las desviaciones estándar de los resultados obtenidos en el experimentos de 6 dimensiones con 4 objetivos y 2 restricciones. Se puede observar que en esta ocasión los resultados de MESMOC, PESMOC y sus versiones desacopladas son diferentes. Por un lado, $\mathrm{PESMOC}_{\mathrm{des}}$ consigue los mejores resultados, tanto cuando hay ruido como cuando no. Esto es gracias a que puede evaluar en el punto de la función objetivo o restricción que más reduce la entropía de la solución, es decir, se puede enfocar en la caja negra que necesite. Sin embargo, la versión desacoplada de MESMOC, presenta un rendimiento inferior al de su contraparte acoplada. No se sabe qué origina el comportamiento de $\mathrm{MESMOC}_{\mathrm{des}}$ para que obtenga estos extraños resultados, aunque se ha visto que esto provoca que al inicio de las ejecuciones (iteraciones 0 - 100), $\mathrm{MESMOC}_{\mathrm{des}}$

devuelva más conjuntos con puntos inválidos que el resto de métodos, lo que provoca que tenga más hipervolumenes con valor 0, así que, su promedio empeora. Por otra parte, MESMOC, PESMOC y BMOO obtienen unos resultados similares en el entorno sin ruido. Sin embargo, cuando sí hay ruido, el desempeño de PESMOC es superior al de los otros dos, aunque MESMOC pasa a obtener mejores resultados que BMOO. Por último, se debe añadir que la búsqueda aleatoria continua siendo la opción menos aconsejable y que más rápido se estanca en los resultados.

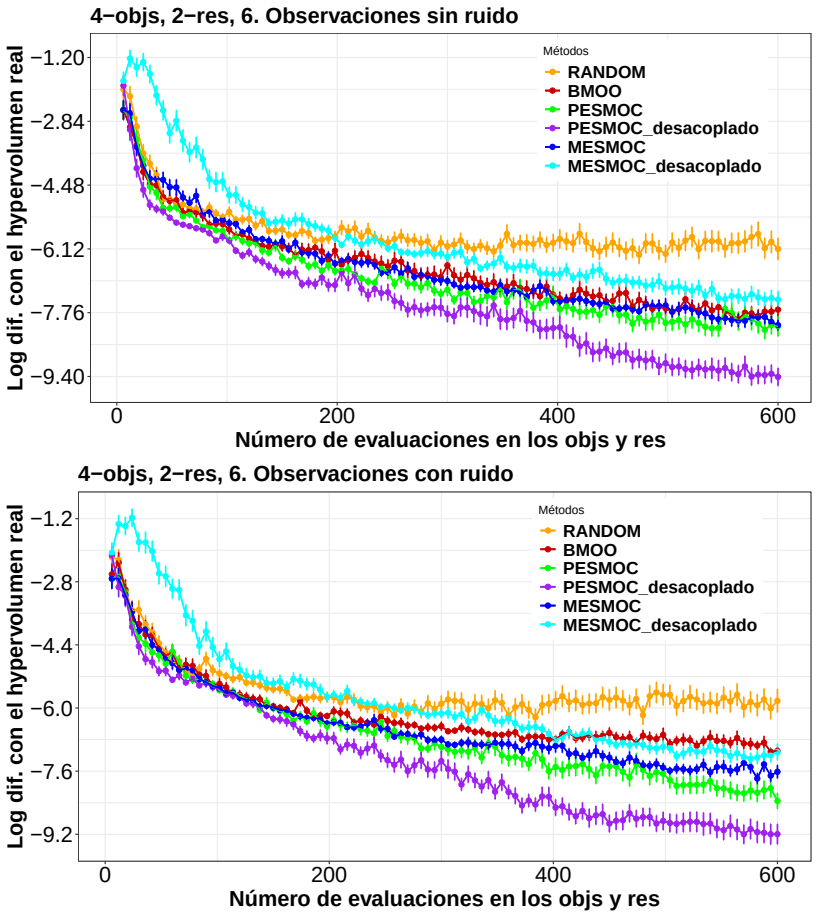


Figura 5.2: Se muestran los resultados del experimento sintético 6d. Al igual que en el experimento 4d, los valores corresponden al promedio de 100 simulaciones de la diferencia en logaritmo del hipervolumen entre los conjuntos de cada método en cada iteración con el conjunto solución.

En la Tabla 5.1 se muestra el tiempo de ejecución que requieren MESMOC, $\text{MESMOC}_{\text{des}}$, PESMOC y $\text{PESMOC}_{\text{des}}$ para completar una iteración. En este tiempo no se incluye la actualización las distribuciones predictivas de los $\mathcal{GP}$s. Se puede observar que MESMOC es la función de adquisición más veloz, seguida de PESMOC. Por otro lado, $\text{MESMOC}_{\text{des}}$ tarda un poco más que PESMOC, pero mucho menos que $\text{PESMOC}_{\text{des}}$. Se debe señalar que la desviación estándar de $\text{PESMOC}_{\text{des}}$ es muy alta, porque la diferencia en el tiempo de ejecución entre las primeras y ultimas evaluaciones es muy elevada, es decir, en las primeras iteraciones $\text{PESMOC}_{\text{des}}$ tarda relativamente poco en comparación con las últimas. Sin embargo, es evidente que MESMOC requiere de menos tiempo de ejecución que PESMOC. Esto se debe a que ejecuta ADF, en lugar de EP, lo que permite que solo haga falta procesar

los factores asociados al punto candidato $\mathbf{x}$, mientras que con EP también habría que procesar los no asociados a $\mathbf{x}$.

Tabla 5.1: Tiempo promedio de ejecución por iteración de MESMOC y PESMOC en el experimento 4d en segundos.

MESMOC	$\mathrm{MESMOC}_{\mathrm{des}}$	PESMOC	$\mathrm{PESMOC}_{\mathrm{des}}$
$43{,}74 \pm 9{,}57$	$129{,}58 \pm 21{,}33$	$101{,}60 \pm 34{,}18$	$421{,}37 \pm 182{,}78$

5.2. Experimentos con datos reales

Se han realizado dos experimentos con datos reales. En ambos casos el objetivo ha sido optimizar los hiperparámetros de un clasificador. En el primero de ellos se ha trabajado en la creación de un conjunto de árboles de decisión y en el segundo en la elaboración de redes neuronales. En los dos experimentos se muestran los resultados para dos cantidades de evaluaciones, a fin de ver el progreso de los clasificadores según aumenta la cantidad de configuraciones de hiperparámetros que pueden probar las diferentes funciones de adquisición.

5.2.1. Experimento con mezclas de árboles de decisión

El primero de los experimentos reales consiste en la optimización de los hiperparámetros de un conjunto de arboles de decisión (ensemble of decision trees). A partir de este punto se utilizará el término ensemble en lugar de conjunto para referirse al conjunto de árboles. La implementación de los árboles de decisión escogida ha sido la que provee Scikit-learn [36]. Por otro lado, este experimento cuenta con dos objetivos y una restricción. Los objetivos son minimizar el error en la clasificación del ensemble y su tamaño. El tamaño del ensemble es la suma del número de nodos de todos sus árboles de decisión. La restricción será explicada más adelante. Por otra parte, el conjunto de datos (dataset) con el que se ha trabajado es el de tarjetas de crédito alemanas (German credit card dataset), y se ha obtenido del repositorio de datasets UCI [37]. Este dataset cuenta con 1000 muestras de datos, con 20 atributos distintos y 2 posibles clases. Por otro lado, para evitar que los valores del error estén sesgados, su cálculo se ha efectuado repitiendo 5 veces una validación cruzada (cross-validation) con 10 divisiones (10-folds).

Al trabajar con ensembles de árboles de decisión, es de vital importancia que haya diversidad entre los árboles, porque si todos son iguales, a la hora de votar (elegir clase para una muestra), todos votarán lo mismo y será como tener un solo árbol. Sin embargo, las elecciones de los árboles no deben ser aleatorias dado que tienen que obtener un acierto superior al $50\,\%$ o de lo contrario el ensemble producto de combinarlos no obtendrá buenos resultados.

Los 5 hiperparámetros elegidos han sido escogidos con la intención de fomentar esta diversidad. El primero de ellos es el número de árboles de decisión a construir, este puede variar entre 1 y 1000. Por otro lado, durante la construcción de los árboles de decisión, en la fase del entrenamiento se dividen los nodos que más ayudan a mejorar la clasificación. La elección de qué nodo dividir se realiza en base a la impureza que reduce dividir ese nodo. Esta impureza se calcula en base a un número de atributos elegidos aleatoriamente de entre los disponibles. La cantidad de atributos a escoger es el segundo de los hiperparámetros a modificar. Al tener 20 atributos, este hiperparámetro puede variar entre 1 y 20. Sin embargo, a partir de cierto punto ya no merece la pena dividir un nodo, porque ese nodo casi no ayuda en la clasificación. El tercero de los hiperparámetros es el número mínimo de muestras con las que trabaja un nodo para dividirlo. Se ha establecido que este hiperparámetro puede variar entre 2 y 200. Por otra parte, también se puede añadir diversidad utilizando una técnica llamada class-switching [38]. Este método consiste en cambiar la clase de algunas muestras del conjunto de entrenamiento con cierta probabilidad. El cuarto hiperparámetro es esta probabilidad, la cual se puede establecer entre $0{,}0$ y $0{,}7$. Por último, también se puede conseguir generar árboles independientes entre ellos si cada uno utiliza unos datos diferentes en el entrenamiento. El quinto hiperparámetro es la fracción de muestras que se utilizan para entrenar los árboles, cuyo intervalo de valores se ha especificado entre $0{,}5$ y $1{,}0$.

Uno de los inconvenientes de trabajar con ensembles es que se debe consultar a cada clasificador, en este caso árboles de decisión, para decidir la clase de una muestra. Sin embargo, es posible acelerar este proceso, dado que a partir de cierto punto no puede cambiar la clase elegida por la mayoría o su probabilidad es muy baja, así que, se deja de preguntar a los clasificadores. Esta técnica se conoce como dynamic ensemble pruning [39]. La restricción utilizada ha consistido en invalidar las observaciones en las que, al utilizar esta técnica, se acelerase menos de un $25\,\%$ el proceso de votación. Se ha elegido esta restricción con la intención de garantizar que esté activa en la solución óptima.

Los objetivos y la restricción elegidas, permiten que este experimento sea válido para ejecutar de forma desacoplada. Para el primero de los objetivos, solamente es necesario contar la cantidad de nodos de los árboles después de su entrenamiento, así que, no hace falta conocer su error. Sin embargo, para el segundo objetivo, se requiere obtener el porcentaje de error en las predicciones, y esto implica realizar todo el proceso de construcción y posterior prueba en el conjunto de validación. Aunque, para la restricción, además de construir el ensemble y validarlo, también se debe elaborar una costosa tabla. Esta tabla se utiliza en la validación e indica la cantidad de votos que tiene que tener la clase mayoritaria dado el número de árboles consultados para finalizar la votación. Además, esta tabla es distinta para cada número de árboles del ensemble.

En la Figura 5.3, se muestra la frontera de Pareto encontrada por cada uno de los métodos utilizados, promediando los resultados de 100 simulaciones, después de realizar 100 y 200 evaluaciones. Tal como se comenta en la Sección 4.1, cuanto mayor sea el hipervolumen que hay por encima de la frontera obtenida por un método, mejor es ese método (asumiendo minimización). Por lo tanto, se

puede observar que los mejores métodos son las variantes desacopladas de MESMOC y PESMOC. Por otro lado, también se puede ver que después de 100 evaluaciones los métodos que encuentran los ensembles con mayor acierto son PESMOC y $\mathrm{PESMOC}_{\mathrm{des}}$, mientras que BMOO obtiene el ensemble que con menor número de nodos reduce el error hasta un $30\,\%$. Asimismo, $\mathrm{MESMOC}_{\mathrm{des}}$ obtiene unos resultados superiores o iguales a $\mathrm{PESMOC}_{\mathrm{des}}$ excepto cuando este último elabora ensembles con menos de un $24\,\%$ de error, los cuales no consigue igualar $\mathrm{MESMOC}_{\mathrm{des}}$. Por otra parte, el desempeño de MESMOC solo se parece al de PESMOC con ensembles si el error es superior al $26\,\%$. En caso contrario, según se reduce el error requerido, sus resultados empeoran poco a poco hasta asemejarse y empeorar los de BMOO y RANDOM. Cuando el número de iteraciones asciende a 200, las ventaja de $\mathrm{MESMOC}_{\mathrm{des}}$ sobre $\mathrm{PESMOC}_{\mathrm{des}}$ se reduce y $\mathrm{PESMOC}_{\mathrm{des}}$ continua obteniendo ensembles con un error más bajo. Por último, se puede apreciar que el incremento en el número de evaluaciones ayuda bastante a MESMOC, dado que sus resultados pasan a ser siempre mejores o iguales a los de BMOO y RANDOM. Excepto para el clasificador más pequeño con un error del $30\,\%$ que continua obteniéndolo BMOO.

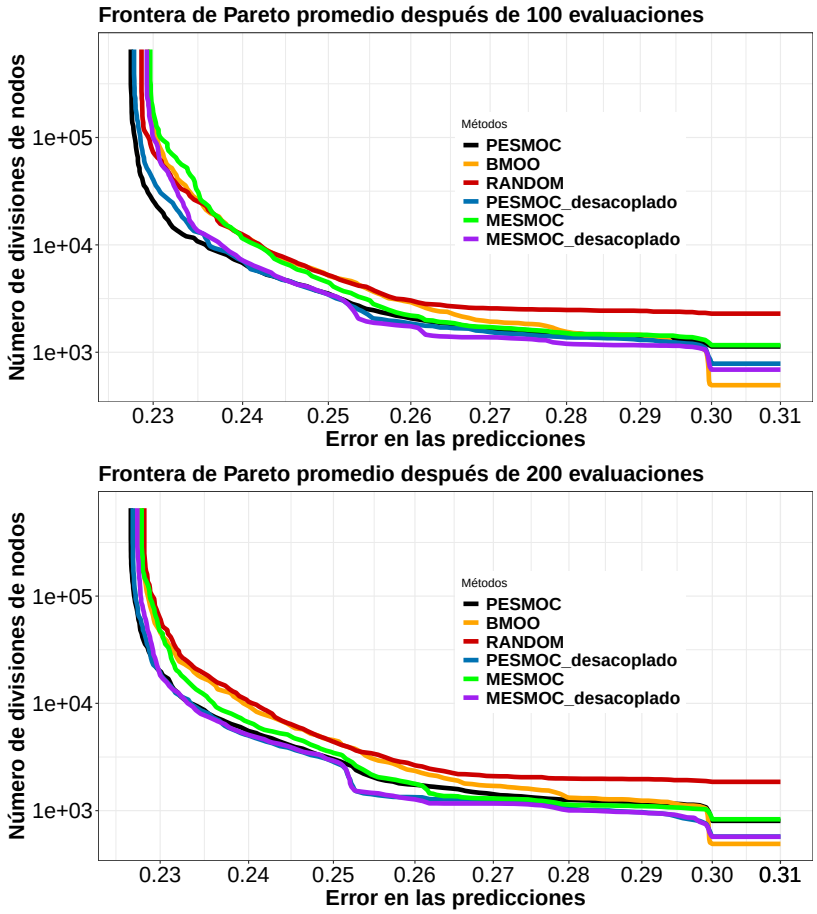


Figura 5.3: Frontera de Pareto obtenida por los diferentes métodos de optimización en el experimento de construir un conjunto de árboles de decisión (ensemble of decision trees). La calidad de los resultados obtenidos se mide en función del tamaño del hipervolumen que hay por encima de los puntos de la frontera (asumiendo minimización).

La Tabla 5.2 presenta el hipervolumen de las fronteras de Pareto que aparecen en la Figura 5.3. Se puede ver que los mejores métodos son aquellos que realizan las evaluaciones de forma desacoplada, dado que les permite enfocarse en las funciones objetivo o restricciones más complicadas, y que por tanto, más reduzcan la entropía de la solución. Aunque el mayor hipervolumen es obtenido por $\mathrm{MESMOC}_{\mathrm{des}}$ apenas hay diferencia con el obtenido por $\mathrm{PESMOC}_{\mathrm{des}}$ después de 200 evaluaciones. Por otro lado, MESMOC es el método más favorecido al aumentar el número de evaluaciones, cuyo resultado pasa a acercarse al de PESMOC.

Tabla 5.2: Promedio del hipervolumen de las fronteras de Pareto de los diferentes métodos para 100 y 200 evaluaciones de las funciones objetivo y restricciones del experimento de ensembles.

Evals	MESMOC	$\mathrm{MESMOC}_{\mathrm{des}}$	PESMOC	$\mathrm{PESMOC}_{\mathrm{des}}$	BMOO	RANDOM
100	$0{,}271 \pm 0{,}016$	$\mathbf{0{,}312 \pm 0{,}011}$	$0{,}297 \pm 0{,}012$	$0{,}307 \pm 0{,}020$	$0{,}287 \pm 0{,}013$	$0{,}236 \pm 0{,}016$
200	$0{,}314 \pm 0{,}011$	$\mathbf{0{,}340 \pm 0{,}007}$	$0{,}323 \pm 0{,}008$	$0{,}339 \pm 0{,}006$	$0{,}305 \pm 0{,}010$	$0{,}256 \pm 0{,}014$

En la Figura 5.4, se muestra el número de evaluaciones realizadas por $\mathrm{MESMOC}_{\mathrm{des}}$ en cada caja negra. Se puede ver que no se enfoca en ninguna en especial, a pesar de que, en principio, la caja negra más difícil de optimizar es el error de predicción. Por lo tanto, parece que para $\mathrm{MESMOC}_{\mathrm{des}}$ todas las cajas negras permiten reducir aproximadamente lo mismo la entropía de la solución del problema.

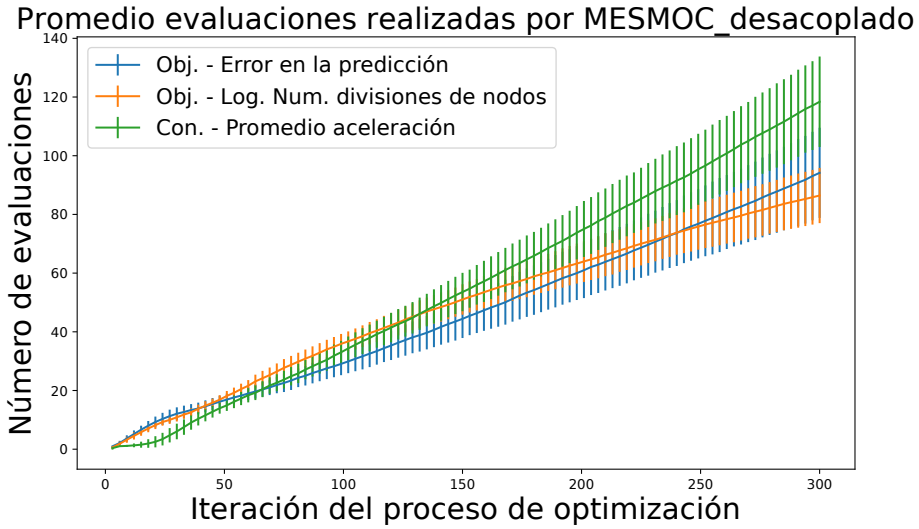


Figura 5.4: Promedio y error estándar del número de evaluaciones que realiza $\mathrm{MESMOC}_{\mathrm{des}}$ en cada una de las cajas negras del experimento con conjuntos de árboles de decisión (ensembles of decision trees) de 100 simulaciones.

5.2.2. Experimento con redes neuronales

En el segundo experimento con datos reales, el objetivo de los métodos es variar los hiperparámetros de una red neuronal profunda para minimizar el tiempo que requieren sus predicciones y su error

cometido en la clasificación. En esta ocasión también se debe cumplir una restricción, que se comenta más adelante. La creación de las redes se ha hecho mediante el envoltorio (wrapper) de Keras para Tensorflow [40, 41], y se ha utilizado el algoritmo ADAM para el entrenamiento de las redes utilizando sus parámetros por defecto [42]. El objetivo de las redes creadas es clasificar imágenes del dataset MNIST [43], el cual contiene $60{,}000$ muestras de 28×28 pixeles de números manuscritos desde el 0 hasta el 9. El dataset se ha dividido en $50{,}000$ muestras para el entrenamiento, y $10{,}000$ la validación. Es posible observar que los dos objetivos son conflictivos dado que redes más grandes obtendrán menor error, pero también necesitarán más tiempo de computo en sus predicciones.

La restricción elegida ha sido que el área requerida de un microchip que almacenase la red neuronal profunda no superase un milímetro cuadrado. El cálculo de este área se realiza mediante el programa de simulación de hardware Aladdin [44]. Este software de simulación permite calcular el área que requiere un programa al ser codificado en un microchip. Se puede apreciar que, al igual que en el experimento con ensembles, en este caso la restricción también ha sido escogida con la intención de que esté activa en los conjuntos de hiperparámetros solución óptimos.

Nuevamente, el experimento permite evaluar los objetivos y las restricciones por separado. Esto es posible dado que se puede medir el tiempo empleado para predecir sin haber entrenado, y por lo tanto sin conocer el error de la red. Para ello, simplemente se necesita construir una red con pesos aleatorios, cronometrar 100 predicciones y calcular su media. Por otro lado, para obtener la estimación del área requerida por su codificación en un microchip tampoco hace falta entrenar la red.

En este experimento se han elegido 8 hiperparámetros posibles a modificar para crear las redes neuronales profundas. El primero es el número de capas ocultas de la red, y el segundo la cantidad de neuronas en cada capa. Por otro lado, como las redes neuronales tienden a sobreajustar (overfitting), también se ha decidido variar el logaritmo del ratio de aprendizaje (learning rate), la probabilidad de tirar los valores de un peso de una capa a la siguiente (dropout) [45] y los regularizadores de pesos ℓ_1 y ℓ_2. Por último, también se han modificado dos hiperparámetros que afectan al tiempo de ejecución del programa y al área del chip, la partición de la memoria y el desenroscado de bucles (loop unrolling). En la Tabla 5.3 se recogen los valores mínimos y máximos para cada hiperparámetro, así como, la cantidad mínima de variación en el valor de cada uno (step).

En la Figura 5.5 se muestran dos gráficas. En la primera, se presentan los resultados obtenidos después de 50 evaluaciones, y en la segunda después de 100. Se puede ver que para 50 evaluaciones, $\mathrm{PESMOC}_{\mathrm{des}}$ obtiene el mayor hipervolumen, seguido de $\mathrm{MESMOC}_{\mathrm{des}}$ y PESMOC. Por otro lado, BMOO obtiene unos resultados parecidos a RANDOM, excepto para un error inferior a $0{,}03$, donde encuentra redes que RANDOM no consigue. Por otra parte, MESMOC y RANDOM obtienen los peores resultados. Cuando aumenta el número de evaluaciones a 100, $\mathrm{MESMOC}_{\mathrm{des}}$ casi obtiene las mismas redes que PESMOC y $\mathrm{PESMOC}_{\mathrm{des}}$ para un error cercano a $0{,}02$, aunque $\mathrm{PESMOC}_{\mathrm{des}}$ mejora las configuraciones que obtiene para otras redes, así que, al final el hipervolumen obtenido por

$\mathrm{PESMOC}_{\mathrm{des}}$ permanece con la misma diferencia del logrado por $\mathrm{MESMOC}_{\mathrm{des}}$ antes del incremento de evaluaciones.

Tabla 5.3: Hiperparámetros usados en la construcción de las redes neuronales. También se exponen los mínimos y máximos valores usados, además, de la variación mínima en cada uno (step).

Hiperparámetro	Min	Max	Step
Nº capas ocultas	1	3	1
Nº neuronas por capa	5	300	1
Learning rate	e^{-200}	1	ϵ
Dropout	0	0.9	ϵ
Regularización ℓ_1	e^{-200}	1	ϵ
Regularización ℓ_2	e^{-200}	1	ϵ
Partición de la memoria	1	32	2^x
Loop unrolling	1	32	2^x

En la Tabla 5.4 se recoge el hipervolumen obtenido para las dos cantidades de evaluaciones. Se puede comprobar como el mayor hipervolumen ha sido obtenido por $\mathrm{PESMOC}_{\mathrm{des}}$ seguido de cerca por $\mathrm{MESMOC}_{\mathrm{des}}$. Aunque la ejecución de las iteraciones de $\mathrm{PESMOC}_{\mathrm{des}}$ es bastante más lenta que la de $\mathrm{MESMOC}_{\mathrm{des}}$, como se muestra en la Tabla 5.1. También es posible apreciar que el mayor incremento de hipervolumen al aumentar el número de evaluaciones es para los métodos acoplados, exceptuando a PESMOC. Esto puede que se deba a que estos métodos no han tenido la posibilidad de realizar suficientes evaluaciones enfocándose en la función objetivo difícil (el error en la predicción), así que al aumentar el número de evaluaciones aumenta más su mejora.

Tabla 5.4: Promedio del hipervolumen de las fronteras de Pareto de los diferentes métodos para 50 y 100 evaluaciones de las funciones objetivo y restricciones del experimento de redes neuronales.

$Evals$	MESMOC	$\mathrm{MESMOC}_{\mathrm{des}}$	PESMOC	$\mathrm{PESMOC}_{\mathrm{des}}$	BMOO	RANDOM
50	$8{,}95 \pm 0{,}398$	$9{,}42 \pm 0{,}288$	$9{,}30 \pm 0{,}351$	$\mathbf{9{,}53 \pm 0{,}234}$	$9{,}07 \pm 0{,}559$	$7{,}37 \pm 1{,}085$
100	$10{,}45 \pm 0{,}571$	$10{,}81 \pm 0{,}270$	$10{,}68 \pm 0{,}331$	$\mathbf{10{,}93 \pm 0{,}311}$	$10{,}50 \pm 0{,}319$	$9{,}15 \pm 0{,}743$

En la Figura 5.6, se muestra el número de evaluaciones realizadas por $\mathrm{MESMOC}_{\mathrm{des}}$ en cada caja negra. Se puede apreciar que en esta ocasión $\mathrm{MESMOC}_{\mathrm{des}}$ se ha enfocado casi por completo en el error en la predicción dejando casi de lado el tiempo requerido para predecir y el área del chip. Es posible que el error en predicción sea mas complicado en comparación con las otras cajas negras en este experimento que en el anterior, y por ello, $\mathrm{MESMOC}_{\mathrm{des}}$ ha destinado más recursos en optimizarlo. Dado este funcionamiento, queda clara la importancia del entorno desacoplado, dado que permite a la función de adquisición enfocarse según sea necesario en cada caja negra, algo que no es posible realizar en el entorno acoplado.

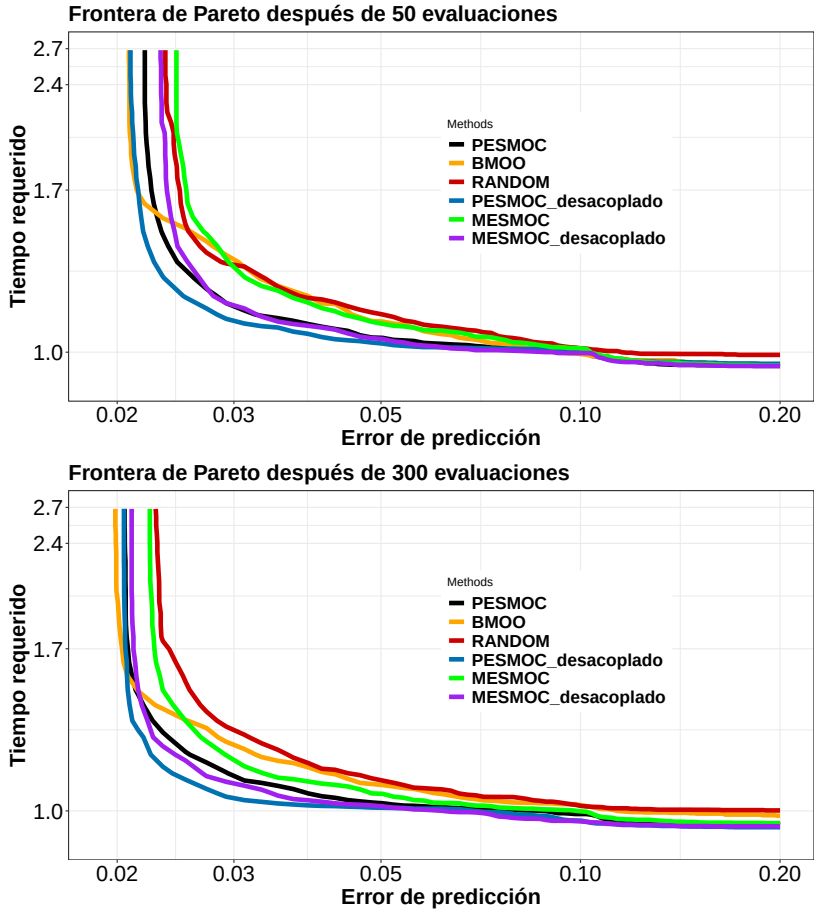


Figura 5.5: Se presenta el hipervolumen obtenido por cada uno de los métodos en el experimento de redes neuronales profundas. El hipervolumen es el área por encima de la frontera de cada método. Cuanto mayor sea este área mejor es el método (asumiendo minimización).

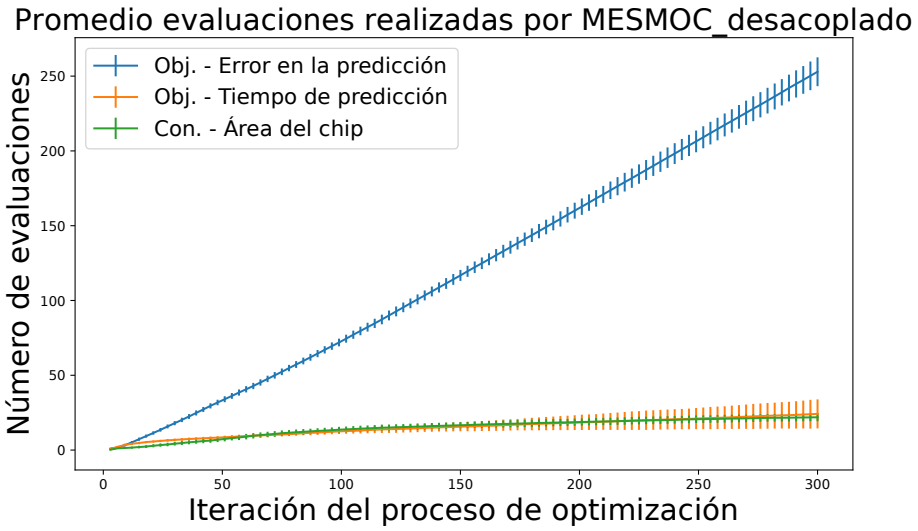


Figura 5.6: Promedio y error estándar del número de evaluaciones que realiza $\mathrm{MESMOC}_{\mathrm{des}}$ en cada una de las cajas negras del experimento con redes neuronales de 100 simulaciones.

6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo se ha explicado como la optimización Bayesiana permite minimizar la cantidad de evaluaciones a realizar en problemas con múltiples objetivos sujetos a varias restricciones. Para lograr este propósito, la optimización Bayesiana dispone de dos piezas elementales. La primera de ellas, es una distribución predictiva (normalmente un proceso Gaussiano) por cada función objetivo o restricción que se quiera modelar. La segunda, es una función de adquisición que evalúa estas distribuciones a fin de obtener cuál sería la siguiente mejor observación a realizar.

También se ha desarrollado MESMOC, una función de adquisición basada en la reducción de la entropía de la solución capaz de afrontar estos problemas multi-objetivo con restricciones desconocidas. Esta función se basa en la entropía del máximo valor al igual que MES, MESC y MESMO, pero estas últimas no pueden afrontar problemas con varios objetivos y restricciones al mismo tiempo. Por otra parte, la derivación de MESMOC ha seguido un enfoque diferente al utilizado por MESMO para trabajar con varios objetivos, dado que los resultados que se obtuvieron con su enfoque no eran los esperados. El tratamiento de varios objetivos de MESMOC es similar al de PESMOC, pero en lugar de aproximar la distribución predictiva condicionada al conjunto de Pareto mediante EP, MESMOC aproxima la distribución predictiva condicionada a la frontera de Pareto usando ADF. Al no utilizar el conjunto de Pareto, sino la frontera, y emplear el método de aproximación ADF en lugar de EP, para las aproximaciones, MESMOC tiene un coste computacional menor a PESMOC, y por tanto es más rápida de ejecutar.

Se realizaron dos experimentos sintéticos y dos reales, en los que se comparó los métodos MESMOC y $\mathrm{MESMOC}_{\mathrm{des}}$ (la variante desacoplada de MESMOC) con PESMOC, $\mathrm{PESMOC}_{\mathrm{des}}$, BMOO, y búsqueda aleatoria. Se observó que $\mathrm{MESMOC}_{\mathrm{des}}$ obtiene un desempeño similar a $\mathrm{PESMOC}_{\mathrm{des}}$ excepto en la prueba con seis cajas negras (4 objetivos y 2 restricciones). Por otro lado, MESMOC solo equiparó los resultados de PESMOC en el experimento sintético 4d y en el 6d sin la presencia de ruido. Por lo tanto, MESMOC es un método con un rendimiento similar o un poco inferior a PESMOC, pero cuya ejecución por iteración es más más rápida. Además, también se ha visto que realizar eva-

luaciones de forma desacoplada puede favorecer sustancialmente el rendimiento, aunque, si la caja negra más complicada, también es la más costosa de evaluar, esto también conlleva un incremento en el coste.

6.2. Trabajo futuro

Es posible encontrar diversas formas de ampliar el trabajo aquí presentado. Por un lado, en algunos escenarios los datos pueden ser valores discretos o categóricos en lugar de continuos, por lo que sería posible extender esta función de adquisición para que pudiera afrontar esos problemas, aplicando un enfoque similar al de [46]. Por otro lado, hay situaciones en las que las restricciones no devuelven un valor, sino que se pueden cumplir o no, así que, se podría añadir esta posibilidad tal como se hace en el desarrollo de MESC [16].

La función de adquisición desarrollada es miope, es decir, solamente considera el impacto que tendrá la siguiente evaluación, sin embargo, si desde el principio se tiene planeado realizar N evaluaciones, lo ideal es que esta función lo tenga en cuenta para guiar la búsqueda de la solución. Para ello, se podría utilizar el enfoque de [47] para extender la función desarrollada en este trabajo.

La velocidad del algoritmo de optimización Bayesiana es crucial, dado que, cuanto más rápido sea, mayor cantidad de problemas podrá afrontar. Por tanto, se hace evidente el uso de alguna técnica para mejorar esta característica. Por un lado, se podría mejorar la eficiencia de ajustar la distribución predictiva, utilizando procesos Gaussianos dispersos (sparse Gaussian processes) [48,49], o se podría emplear un modelo probabilístico basado en redes neuronales que permite paralelizarlo [29]. También sería posible modificar la función de adquisición para un entorno paralelo, de forma que, en cada iteración recomiende un conjunto de puntos donde evaluar en lugar de uno solo, tal como se hace en [50] para PESMOC.

También existen problemas donde es necesario utilizar una distribución predictiva con mayor flexibilidad que la ofrecida por los procesos Gaussianos convencionales. En estos casos se podría utilizar procesos Gaussianos profundos [51, 52]. Asimismo, es posible que la aproximación por ADF sea demasiado sencilla, así que, se podría utilizar en su lugar EP.

Bibliografía

[1] E. Brochu, V. M. Cora, and N. de Freitas, *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. Technical Report TR-2009-023, University of British Columbia, 2009. Descarga.

[2] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015. Descarga.

[3] Y. Collette and P. Siarry, *Multiobjective optimization: principles and case studies*. Springer Science & Business Media, 2013. Descarga.

[4] P. Feliot, J. Bect, and E. Vazquez, "A bayesian approach to constrained single-and multi-objective optimization," *Journal of Global Optimization*, vol. 67, no. 1-2, pp. 97–133, 2017. Descarga.

[5] E. C. Garrido-Merchán and D. Hernández-Lobato, "Predictive entropy search for multi-objective bayesian optimization with constraints," *Neurocomputing*, vol. 361, pp. 50–68, 2019. Descarga.

[6] H. J. Kushner, "A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise," *J. Basic Engineering*, vol. 86, pp. 87–106, 1964.

[7] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of bayesian methods for seeking the extremum," *Towards global optimization*, vol. 2, no. 117-129, p. 2, 1978.

[8] D. D. Cox and S. John, *A statistical method for global optimization*. IEEE, 1992. Descarga.

[9] V. Picheny, "Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction," *Statistics and Computing*, vol. 25, no. 6, pp. 1265–1280, 2015. Descarga.

[10] M. Emmerich and J. W. Klinkenberg, *The computation of the expected improvement in dominated hypervolume of Pareto front approximations*. Rapport technique, Leiden University, 2008. Descarga.

[11] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection," in *Parallel Problem Solving from Nature – PPSN X*, pp. 784–794, Springer Berlin Heidelberg, 2008. Descarga.

[12] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints.," in *Proceedings of the International Conference on Machine Learning*, pp. 937–945, 2014. Descarga.

[13] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," *Proceedings of the Uncertainty in Artificial Intelligence*, 2014. Descarga.

[14] J. R. Snoek, *Bayesian optimization and semiparametric models with applications to assistive technology*. PhD thesis, University of Toronto, 2013. Descarga.

[15] D. Hernández-Lobato, J. M. Hernández-Lobato, A. Shah, and R. Adams, "Predictive entropy search for multi-objective bayesian optimization," *International Conference on Machine Learning*,

pp. 1492–1501, 2016. Descarga.

[16] J. M. Hernández-Lobato, M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani, "Predictive entropy search for bayesian optimization with unknown constraints," *International conference on machine learning*, pp. 1699–1707, 2015. Descarga.

[17] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, "Predictive entropy search for efficient global optimization of black-box functions," *Advances in neural information processing systems*, pp. 918–926, 2014. Descarga.

[18] Z. Wang and S. Jegelka, "Max-value entropy search for efficient bayesian optimization," *International Conference on Machine Learning 34*, pp. 3627–3635, 2017. Descarga.

[19] S. Belakaria, A. Deshwal, and J. R. Doppa, "Max-value entropy search for multi-objective bayesian optimization," *Advances in Neural Information Processing Systems*, pp. 7823–7833, 2019. Descarga.

[20] V. Perrone, I. Shcherbatyi, R. Jenatton, C. Archambeau, and M. Seeger, "Constrained bayesian optimization with max-value entropy search," 2019. Descarga.

[21] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006. Descarga.

[22] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2006. Descarga.

[23] B. Matern, *Reports of the Forest Research Institute of Sweden*. Springer-Verlang, 1960.

[24] M. Abramowitz and I. A. Stegun, "Handbook of mathematical functions. 1965," 1964.

[25] I. Murray and R. P. Adams, "Slice sampling covariance hyperparameters of latent gaussian models," *Advances in neural information processing systems*, pp. 1732–1740, 2010. Descarga.

[26] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, pp. 2951–2959, 2012. Descarga.

[27] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization," *Journal of Machine Learning Research*, vol. 13, no. Jun, pp. 1809–1837, 2012. Descarga.

[28] A. Shah, A. Wilson, and Z. Ghahramani, "Student-t processes as alternatives to gaussian processes," in *Artificial intelligence and statistics*, pp. 877–885, 2014. Descarga.

[29] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, Prabhat, and R. Adams, "Scalable Bayesian optimization using deep neural networks," *International Conference on Machine Learning*, 2015. Descarga.

[30] J. Knowles, "Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006. Descarga.

[31] M. Schonlau, W. J. Welch, and D. R. Jones, *Global versus local search in constrained optimization of computer models*, vol. Volume 34 of *Lecture Notes–Monograph Series*. Hayward, CA: Institute of Mathematical Statistics, 1998. Descarga.

[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

Descarga.

[33] T. P. Minka, “Expectation propagation for approximate bayesian inference,” *Proceedings of the Uncertainty in Artificial Intelligence*, pp. 362–369, 2001. Descarga.

[34] X. Boyen and D. Koller, “Tractable inference for complex stochastic processes,” *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 33–42, 1998. Descarga.

[35] D. Hernández-Lobato, *Prediction Based on Averages over Automatically Induced Learners: Ensemble Methods and Bayesian Techniques*. PhD thesis, Universidad Autónoma de Madrid, 2010. Descarga.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. Descarga.

[37] A. Asuncion and D. Newman, “UCI machine learning repository,” 2007. Enlace.

[38] G. Martínez-Muñoz and A. Suárez, “Switching class labels to generate classification ensembles,” *Pattern Recognition*, vol. 38, no. 10, pp. 1483–1494, 2005. Descarga.

[39] D. Hernández-Lobato, G. Martínez-Muñoz, and A. Suárez, “Statistical instance-based pruning in ensembles of independent classifiers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 364–369, 2008. Descarga.

[40] F. Chollet *et al.*, “Keras: The python deep learning library,” *ascl*, pp. ascl–1806, 2018. Enlace.

[41] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. Software available from tensorflow.org.

[42] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *Proceedings of the International Conference on Learning Representations*, 2014. Descarga.

[43] Y. LeCun, C. Cortes, and C. J. Burges, “MNIST handwritten digit database, 2,” 2010. Enlace.

[44] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, “Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures,” in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pp. 97–108, IEEE, 2014. Descarga.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. Descarga.

[46] E. C. Garrido-Merchán and D. Hernández-Lobato, “Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes,” *Neurocomputing*, vol. 380, pp. 20–35, 2020. Descarga.

[47] J. González, M. Osborne, and N. Lawrence, “Glasses: Relieving the myopia of bayesian optimisation,” in *Artificial Intelligence and Statistics*, pp. 790–799, 2016. Descarga.

[48] E. Snelson and Z. Ghahramani, “Sparse gaussian processes using pseudo-inputs,” *In Advances in Neural Information Processing Systems*, no. 5, pp. 1257–1264, 2005. Descarga.

[49] J. Quiñonero-Candela and C. E. Rasmussen, “A unifying view of sparse approximate gaussian process regression,” *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005. Descarga.

[50] E. C. Garrido-Merchán and D. Hernández-Lobato, “Parallel predictive entropy search for multi-objective bayesian optimization with constraints,” *Bayesian Inference In Stochastic Processes*, 2020. Descarga.

[51] A. Damianou and N. Lawrence, “Deep gaussian processes,” in *Artificial Intelligence and Statistics*, pp. 207–215, 2013. Descarga.

[52] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, and R. Turner, “Deep gaussian processes for regression using approximate expectation propagation,” in *International conference on machine learning*, pp. 1472–1481, 2016. Descarga.

Apéndices

A

Apéndice del Capítulo 4

A continuación se muestran los cálculos realizados para las derivadas $\frac{\partial \log(Z)}{\partial m_j^f}$, $\frac{\partial \log(Z)}{\partial v_j^f}$, $\frac{\partial \log(Z)}{\partial m_j^c}$ y $\frac{\partial \log(Z)}{\partial v_j^c}$ necesarias para calcular ADF:

$$
\begin{aligned}
\frac{\partial \log(Z)}{\partial m_j^f} &= \frac{\partial}{\partial m_j^f} \log\left(1 - \prod_{i=1}^{C} \Phi(\gamma_i^c) \prod_{j=1}^{K} \Phi(\gamma_j^f)\right) \\
&= \frac{1}{Z} \frac{\partial}{\partial m_j^f} \left(1 - \prod_{i=1}^{C} \Phi(\gamma_i^c) \prod_{j=1}^{K} \Phi(\gamma_j^f)\right) \\
&= \frac{1}{Z} \left(-\frac{\partial}{\partial m_j^f} \left(\Phi\left(\frac{f^\star - m_j^f}{(v_j^f)^{1/2}}\right)\right) \prod_{i=1}^{C} \Phi(\gamma_i^c) \prod_{k\neq j}^{K} \Phi(\gamma_k^f)\right) \\
&= \frac{1}{Z} \left(-\left(\mathcal{N}(\gamma_j^f) \frac{-1}{(v_j^f)^{1/2}}\right) \prod_{i=1}^{C} \Phi(\gamma_i^c) \prod_{k\neq j}^{K} \Phi(\gamma_k^f)\right) \\
&= \frac{1}{Z} \left(-\prod_{i=1}^{C} \Phi(\gamma_i^c) \prod_{k\neq j}^{K} \Phi(\gamma_k^f)\right) \left(\mathcal{N}(\gamma_j^f) \frac{-1}{(v_j^f)^{1/2}}\right) \\
&= \frac{(Z-1)}{Z\Phi(\gamma_j^f)} \left(\frac{-\mathcal{N}(\gamma_j^f)}{(v_j^f)^{1/2}}\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \log(Z)}{\partial m_j^c} &= \frac{1}{Z} \left(-\frac{\partial}{\partial m_j^c} \left(\Phi\left(\frac{m_j^c}{(v_j^c)^{1/2}}\right)\right) \prod_{i\neq j}^{C} \Phi(\gamma_i^c) \prod_{k=1}^{K} \Phi(\gamma_k^f)\right) \\
&= \frac{1}{Z} \left(-\left(\mathcal{N}(\gamma_j^c) \frac{1}{(v_j^c)^{1/2}}\right) \prod_{i\neq j}^{C} \Phi(\gamma_i^c) \prod_{k=1}^{K} \Phi(\gamma_k^f)\right) \\
&= \frac{1}{Z} \left(-\prod_{i\neq j}^{C} \Phi(\gamma_i^c) \prod_{k=1}^{K} \Phi(\gamma_k^f)\right) \left(\mathcal{N}(\gamma_j^c) \frac{1}{(v_j^c)^{1/2}}\right) \\
&= \frac{(Z-1)}{Z\Phi(\gamma_j^c)} \left(\frac{\mathcal{N}(\gamma_j^c)}{(v_j^c)^{1/2}}\right)
\end{aligned}
$$

$$\begin{aligned}
\frac{\partial \log(Z)}{\partial v_j^f} &= \frac{1}{Z}\left(-\frac{\partial}{\partial v_j^f}\left(\Phi\left(\frac{f_j^\star - m_j^f}{(v_j^f)^{1/2}}\right)\right)\prod_{i=1}^{C}\Phi(\gamma_i^c)\prod_{k\neq j}^{K}\Phi(\gamma_k^f)\right) \\
&= \frac{1}{Z}\left(-\left(-\mathcal{N}(\gamma_j^f)\frac{f_j^\star - m_j^f}{2(v_j^f)^{3/2}}\right)\prod_{i=1}^{C}\Phi(\gamma_i^c)\prod_{k\neq j}^{K}\Phi(\gamma_k^f)\right) \\
&= \frac{1}{Z}\left(-\prod_{i=1}^{C}\Phi(\gamma_i^c)\prod_{k\neq j}^{K}\Phi(\gamma_k^f)\right)\left(\mathcal{N}(\gamma_j^f)\frac{-\gamma_j^f}{2v_j^f}\right) \\
&= \frac{(Z-1)}{Z\Phi(\gamma_j^f)}\left(\mathcal{N}(\gamma_j^f)\frac{-\gamma_j^f}{2v_j^f}\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \log(Z)}{\partial v_j^c} &= \frac{1}{Z}\left(-\frac{\partial}{\partial v_j^c}\left(\Phi\left(\frac{m_j^c}{(v_j^c)^{1/2}}\right)\right)\prod_{i\neq j}^{C}\Phi(\gamma_i^c)\prod_{k=1}^{K}\Phi(\gamma_k^f)\right) \\
&= \frac{1}{Z}\left(-\left(-\mathcal{N}(\gamma_j^f)\frac{m_j^c}{2(v_j^c)^{3/2}}\right)\prod_{i\neq j}^{C}\Phi(\gamma_i^c)\prod_{k=1}^{K}\Phi(\gamma_k^f)\right) \\
&= \frac{1}{Z}\left(-\prod_{i\neq j}^{C}\Phi(\gamma_i^c)\prod_{k=1}^{K}\Phi(\gamma_k^f)\right)\left(\mathcal{N}(\gamma_j^f)\frac{-\gamma_j^c}{2v_j^c}\right) \\
&= \frac{(Z-1)}{Z\Phi(\gamma_j^c)}\left(\mathcal{N}(\gamma_j^c)\frac{-\gamma_j^c}{2v_j^c}\right)
\end{aligned}$$

donde $\Phi(\cdot)$ es la distribución de probabilidad acumulada de una Gaussiana estándar y $\mathcal{N}$ es la función de densidad de probabilidad de una Gaussiana estándar.