Vers une approche floue pour le design de plan expérimental

Olivier Rousselle, Jean-Philippe Poli, Nadia Ben Abdallah

Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France

{prenom.nom}@cea.fr

Résumé

Nous présentons dans cet article une approche floue interprétable du design expérimental sous contraintes qui peut être utilisée avec peu de données. L'objectif est de fournir aux expérimentateurs un algorithme leur permettant d'échantillonner de manière optimale. Nous détaillons les différentes étapes de notre algorithme qui consiste à recommander la prochaine expérience à réaliser et à construire une base de règles floues de Sugeno. Nous présentons ensuite quelques résultats de notre algorithme, que nous comparons avec l'approche bayésienne.

Mots-clés

Apprentissage actif, design expérimental, flou, règles, optimisation, interprétabilité.

Abstract

In this paper we present an interpretable fuzzy approach to constrained experimental design that can be used with little data. The objective is to provide experimenters with an algorithm allowing them to sample optimally. We detail the different steps of our algorithm which consists of recommending the next experiment to be carried out and constructing a Sugeno fuzzy rule base. We will then present some results of our algorithm, which we compare with the Bayesian approach.

Keywords

Active learning, experimental design, fuzzy, rules, optimization, interpretability.

1 Introduction

Les sciences expérimentales nécessitent d'explorer un espace de possibilités souvent très vaste pour s'approcher d'un optimum. Classiquement, l'approche par essais et erreurs est utilisée dans ce domaine pour collecter des données expérimentales, dont la production peut être coûteuse et longue. Le processus est répété jusqu'à ce qu'une propriété ou une performance désirée soit atteinte.

Différentes méthodes d'échantillonnage sont utilisées dans la recherche expérimentale, telles que l'échantillonnage aléatoire, l'échantillonnage factoriel, la méthode des surfaces de réponse, l'optimisation bayésienne [1], l'algorithme de couverture optimale [2], etc. Les réseaux de neurones ont également été utilisés pour la recherche expérimentale [3], mais ont l'inconvénient d'être une boîte noire.

Sans perte de généralité, nous prenons comme exemple la découverte des matériaux, qui vise à produire des matériaux performants pour un usage ciblé. Ces matériaux sont généralement produits à partir d'un mélange de composés initiaux soumis à un certain procédé de fabrication. Ces dernières années, l'intelligence artificielle a accéléré l'innovation dans ces domaines [4, 5, 6].

Dans ce contexte, l'objectif de nos travaux est de développer une méthode basée sur la logique floue appliquée au plan expérimental. Nous définissons notre problème comme tester différents ensembles de paramètres (c'est-à-dire la composition d'un matériau et les paramètres du procédé) pour maximiser une propriété donnée (par exemple la robustesse de ce matériau). En particulier, nous souhaitons trouver une méthode automatique pour échantillonner de manière itérative et optimale les paramètres expérimentaux.

Notre motivation est de fournir un outil pour aider les expérimentateurs à déterminer quels sont les prochains ensembles de paramètres à tester et qui fonctionne avec peu de données. Nous visons à réduire le nombre d'expérimentations pour atteindre une performance cible, à la fois pour réduire le gaspillage de matières premières et converger plus rapidement vers un matériau innovant. Pour garder l'expert humain dans la boucle, nous prêtons attention à l'interprétabilité, en donnant des indices à l'utilisateur sur le choix de la prochaine configuration expérimentale. En effet, l'explicabilité/interprétabilité vise à rendre le fonctionnement et les résultats du modèle plus intelligibles et transparents pour les humains, afin de renforcer la confiance dans la prise de décision et ainsi son acceptabilité.

Le document est structuré comme suit. La section suivante donne un aperçu de l'approche. La section 3 décrit la méthode de régression qui se rapproche de la fonction objectif. La section 4 explique le processus derrière la sélection de la prochaine expérience à réaliser. Nous montrons les résultats et la comparaison avec l'optimisation bayésienne dans la Section 6. Notre approche étant dédiée aux experts humains, la Section 7 présente la manière dont l'utilisateur final est considéré. Enfin, nous tirons quelques conclusions et perspectives.

2 Vue d'ensemble de l'approche

Pour satisfaire les besoins des expérimentateurs, nous avons conçu une approche basée les principes suivants :

— Elle doit mettre en œuvre un échantillonnage adap-

tatif [7], c'est-à-dire un processus séquentiel qui décide de l'emplacement du point suivant en équilibrant les critères d'exploration et d'exploitation ;
— Elle doit être capable de combiner apprentissage (à partir de quelques données expérimentales) et modélisation de connaissances expertes ;
— Robustesse : de petits changements dans les points initiaux ne doivent pas entraîner de changements importants dans les résultats et les prédictions ;
— Interprétabilité : les étapes et les résultats du modèle doivent être intelligibles pour les humains, pour renforcer la confiance dans la prise de décision ;
— Notre approche doit pouvoir travailler sur des problèmes de mélanges de grande dimension.

Pour répondre à ces prérequis, nous avons développé une approche dont les différentes étapes sont détaillées dans la Fig. 1.

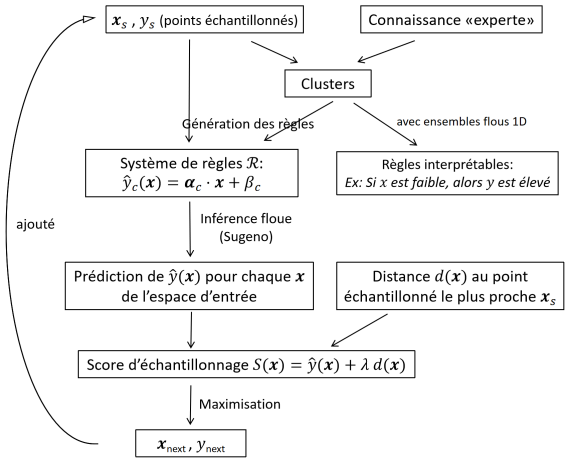

FIGURE 1 – Différentes étapes de l'approche proposée.

Le point expérimental sélectionné est celui qui maximise le score d'échantillonnage (section 4). Ce processus est répété sur plusieurs itérations, chaque itération correspondant à une expérience.

Dans ce travail, nous considérons que les propriétés des matériaux sont données sous forme de valeurs réelles (souvent bornées). Nous devons donc construire une base de règles floues pour une régression (voir section 3). Nous avons choisi d'utiliser un système d'inférence flou de type Sugeno pour son efficacité et pour le fait qu'il fonctionne avec les techniques d'optimisation adaptatives.

Avec ce choix, nous avons privilégié les performances de la prédiction plutôt que l'interprétabilité. Pour compenser, nous proposons une méthode pour extraire un substitut plus interprétable du modèle (section 5).

Remarque : les notations utilisées dans cet article sont détaillées en annexe.

3 Algorithme de régression basé sur le clustering flou

Considérons le problème de la prédiction de la propriété d'un matériau, $\hat{y}$, pour chaque point de l'espace d'entrée (c'est-à-dire tout mélange de matières premières et de paramètres de processus). Nous avons basé notre approche sur plusieurs travaux antérieurs [8, 9, 10] qui partagent l'utilisation d'une méthode de clustering comme première étape dans l'induction de règles. Notre méthode diffère légèrement dans le sens où elle est destinée à des problèmes éventuellement de grande dimension. De plus, nous avons amélioré la manière dont les fonctions d'appartenance sont apprises et le calcul des coefficients de régression pour les conclusions de la règle de Sugeno.

3.1 Clustering

Tout d'abord, nous effectuons un clustering flou multidimensionnel des points déjà échantillonnés $\boldsymbol{x}_s$ pour mettre en évidence différents groupes. Rappelons que la particularité du clustering flou est qu'un point peut appartenir à plusieurs clusters, avec éventuellement des degrés d'appartenance différents.

Le clustering flou s'applique à la fois aux variables d'entrée et de sortie, et chaque cluster c comprend un centre noté $\boldsymbol{m}_c$. Nous notons C l'ensemble des clusters. Le nombre de clusters n_c est souvent un hyperparamètre, dont la valeur doit être définie en respectant les considérations suivantes :

— Trop de clusters peuvent conduire à un surapprentissage. Le modèle s'est bien adapté aux données d'entraînement (points déjà échantillonnés), mais il peut avoir du mal à se généraliser à de nouvelles données. De plus, avoir trop de clusters implique trop de paramètres, et l'optimisation décrite plus loin dans cet article ne sera pas possible.
— Un faible nombre de clusters peut faciliter l'interprétation du modèle et réduire la complexité des calculs (moins de paramètres à optimiser).

Le nombre de clusters peut rester constant au cours des différentes itérations de l'expérience ou peut augmenter régulièrement par paliers en fonction du nombre de points déjà échantillonnés.

Nous nous intéressons maintenant à mesurer le degré d'appartenance d'un point donné $\boldsymbol{x}$ dans l'espace d'entrée à chaque cluster, noté $\mu_c(\boldsymbol{x})$. Nous avons choisi de construire une fonction d'appartenance qui dépend de plusieurs variables d'entrée. L'avantage est de prendre en compte l'interaction entre les variables et d'obtenir une partition forte multidimensionnelle :

$$\forall \boldsymbol{x}\ ,\ \sum_{c \in C} \mu_c(\boldsymbol{x}) = 1. \tag{1}$$

Les degrés d'appartenance peuvent être calculés en s'inspirant de l'algorithme FCM (Fuzzy Clustering Means) [11], c'est-à-dire en minimisant la fonction objectif [12] :

$$\sum_{\boldsymbol{x}_s} \sum_{\boldsymbol{m}_c} \mu_c(\boldsymbol{x}_s) ||\boldsymbol{x}_s - \boldsymbol{m}_c||^2 \tag{2}$$

avec $\boldsymbol{x}_s$ les points considérés, m_c les centres de chaque cluster, et $\mu_c(\boldsymbol{x}_s)$ le coefficient d'appartenance du point $\boldsymbol{x}_s$ au cluster c. Les degrés d'appartenance obtenus sont :

$$\mu_c(\boldsymbol{x}_s) = \sum_{c'} \left(\frac{||\boldsymbol{x}_s - \boldsymbol{m}_c||^2}{||\boldsymbol{x}_s - \boldsymbol{m}_{c'}||^2} \right)^{-\frac{2}{m-1}} \tag{3}$$

avec m le paramètre "fuzzifier" qui influence le flou de la partition. Il va de 1 (partition nette) à $+\infty$. Généralement, m est choisi égal à 2. Chaque point $\boldsymbol{x}$ se voit ensuite attribuer un degré d'appartenance à chaque cluster. Des exemples de degrés d'appartenance sont illustrés dans le diagramme ternaire sur la Fig. 2 avec 3 clusters et avec 3 variables d'entrée x_1, x_2, x_3. En guise de lecture du diagramme ternaire, la croix rouge indique par exemple le point $(x_1, x_2, x_3) = (0.55, 0.2, 0.25)$.

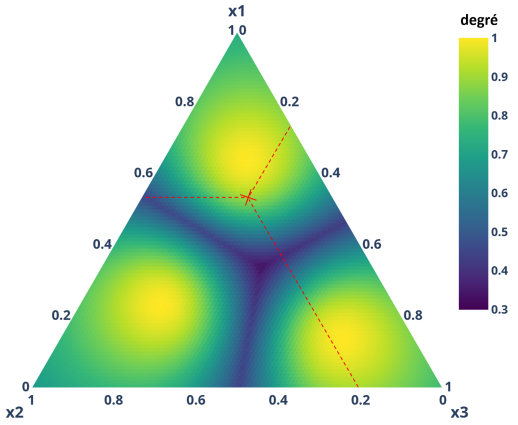

FIGURE 2 – Degrés d'appartenance - étude de cas avec 3 variables de composition et 3 clusters.

La méthode ne vise pas à caractériser les clusters, mais à les décrire dans leur ensemble. La distribution associée correspond donc à des degrés d'appartenance qui indiquent dans quelle mesure un point appartient à chaque cluster [13].
Remarque : on peut également construire une fonction d'appartenance qui dépend uniquement de la distance au centre du cluster. Il peut prendre une forme triangulaire, gaussienne [8, 14] ou Cauchy [10]. Cependant, nous n'avons pas choisi une telle fonction d'appartenance car elle ne considère que les similarités internes au sein d'un cluster (et non les dissemblances externes avec d'autres clusters), et parce que la partition floue n'est pas forte (la somme des degrés d'appartenance n'est pas égale à 1).

3.2 Génération de règles floues

Chaque cluster/région c est à l'origine d'une règle R. Chaque règle R est caractérisée par sa région antécédente multidimensionnelle et ses coefficients de régression $(\boldsymbol{\alpha}_c, \beta_c)$, sous la forme :

$$\hat{y}_c(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_{c_i} x_i + \beta_c = \boldsymbol{\alpha}_c \cdot \boldsymbol{x} + \beta_c. \tag{4}$$

Les coefficients $(\boldsymbol{\alpha}_c, \beta_c)$ de chaque règle sont déterminés par le processus d'optimisation décrit plus loin dans cet article.

Règle experte Une règle peut également provenir d'une expertise humaine ou être issue de la littérature. Par exemple, un expert peut indiquer une autre région d'intérêt en ajoutant un autre centre $\boldsymbol{m}_c$. Ce cluster générera également une règle caractérisée par sa région antécédente et ses coefficients de régression $(\boldsymbol{\alpha}_c, \beta_c)$. Cette règle sera ensuite ajoutée au système de règles déjà généré à partir des données.

Cas des variables discrètes Dans le cas de variables discrètes ou catégorielles, il est nécessaire de transformer ces données pour pouvoir intégrer ces variables dans la régression. Pour ce faire, nous utilisons l'encodage One-Hot, qui consiste à transformer la variable en plusieurs variables binaires, où chaque variable binaire représente une catégorie unique de la variable d'origine.

Processus récursif pour le partitionnement flou A chaque nouveau point échantillonné, nous évaluons à quel cluster il appartient, c'est-à-dire le cluster ayant le plus haut degré d'appartenance. Le centre $\boldsymbol{m}_c$ du cluster c est ensuite modifié en calculant la moyenne pondérée suivante (en notant $\boldsymbol{x}_{last}$ le dernier point testé) [15] :

$$\boldsymbol{m}_c' = \frac{\boldsymbol{m}_c + \mu_c(\boldsymbol{x}_{last})\, \boldsymbol{x}_{last}}{\mu_c(\boldsymbol{x}_{last})}. \tag{5}$$

3.3 Optimisation des coefficients de régression

Le résultat de notre modèle est généré en combinant les fonctions linéaires des règles, comme dans les systèmes de Sugeno :

$$\hat{y}(\boldsymbol{x}) = \sum_{c} \mu_c(\boldsymbol{x})\, \hat{y}_c(\boldsymbol{x}) \tag{6}$$

avec $\hat{y}_c(\boldsymbol{x}) = \boldsymbol{\alpha}_c \cdot \boldsymbol{x} + \beta_c$.
Nous déterminons les coefficients de régression optimaux $(\boldsymbol{\alpha}_c, \beta_c)$ en minimisant l'écart au carré entre les valeurs prédites des points déjà échantillonnés $\hat{y}$ et leurs valeurs réelles y . Chaque cluster contient $N + 1$ coefficients à optimiser. Au total, il y a donc $n_c(N + 1)$ paramètres à optimiser, il nous faut donc au moins $n_c(N + 1)$ points pour réaliser cette optimisation.
Pour chaque point déjà échantillonné, nous prédisons la valeur de sortie à l'aide de la formule d'inférence, que nous comparons à la valeur de sortie réelle. L'objectif est de minimiser la différence entre les valeurs prédites $\hat{y}$ et les valeurs réelles y. On cherche donc à minimiser la fonction de perte :

$$\begin{aligned} L(\boldsymbol{\alpha}, \beta) &= \sum_{\boldsymbol{x}_s} \left(\hat{y}(\boldsymbol{x}_s) - y(\boldsymbol{x}_s)\right)^2 \\ &= \sum_{\boldsymbol{x}_s} \left(\sum_{c} \mu_c(\boldsymbol{x}_s)\, (\boldsymbol{\alpha}_c \cdot \boldsymbol{x}_s + \beta_c) - y(\boldsymbol{x}_s) \right)^2. \end{aligned} \tag{7}$$

3.4 Prédiction de la valeur de sortie pour chaque point d'entrée

Pour chaque point d'entrée $\boldsymbol{x}$, nous prédisons la valeur de sortie $\hat{y}$. La figure 3 montre un exemple de valeurs prédites $\hat{y}$ obtenues avec notre algorithme pour un cas avec 3 variables d'entrée.
Nous pouvons évaluer la précision du modèle d'inférence en utilisant les points déjà échantillonnés. Pour chaque

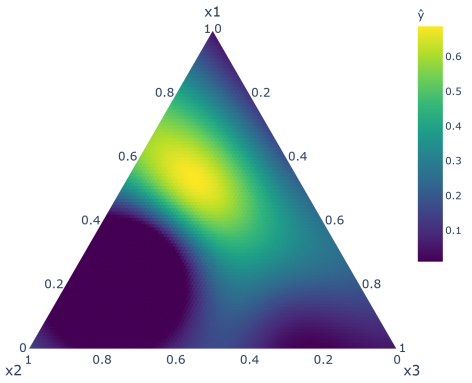

FIGURE 3 – Diagramme ternaire illustrant un exemple de valeurs prédites $\hat{y}$ pour un cas avec 3 variables d'entrée.

point échantillonné, nous prédisons la valeur de sortie $\hat{y}$ à l'aide du système d'inférence, que nous comparons à la valeur réelle y. On peut alors calculer l'écart au carré moyen RMSE et le coefficient de détermination R^2 pour juger de la qualité du modèle. L'objectif est d'avoir un modèle qui maximise R^2 et minimise le RMSE. Les résultats sont présentés dans la partie 6.

Dans [10], les auteurs ont montré que l'algorithme de régression basé sur le clustering flou avec descente de gradient est plus efficace que les réseaux de neurones pour un problème de régression (prédiction de la forme d'une fonction) pour le cas 1D, et avec beaucoup moins d'hyperparamètres.

4 Sélection de la prochaine expérience

La méthode proposée choisit également le prochain point à échantillonner de manière déterministe, comme un compromis entre exploitation et exploration. L'exploitation signifie privilégier les régions à fort potentiel, c'est-à-dire avec des valeurs de sortie élevées prédites $\hat{y}$, tandis que l'exploration signifie explorer les régions encore non testées.

Nous devons introduire une variable qui capture cette partie exploration. Une variable naturelle pour cela est la distance euclidienne au point échantillonné le plus proche, notée d. Le calcul des distances entre chaque point de l'espace d'entrée et le point déjà échantillonné le plus proche est effectué de manière optimisée à l'aide de l'algorithme KD-tree [16]. Ensuite, pour chaque point d'entrée $\boldsymbol{x}$, $\hat{y}$ et d sont calculés ; $\hat{y}$ code l'exploitation, et d code l'exploration. Pour trouver un compromis entre exploitation et exploration, nous introduisons une nouvelle variable appelée "score d'échantillonnage" et notée S :

$$S(\boldsymbol{x}) = \hat{y}(\boldsymbol{x}) + \lambda\, d(\boldsymbol{x}) \tag{8}$$

où $\hat{y}$ et d sont ici normalisés, et où λ est un hyperparamètre. λ peut être choisi constant ou il peut changer en fonction du nombre d'expériences réalisées. Augmenter λ favorisera l'exploration par rapport à l'exploitation.

Une illustration de S est présentée dans la Fig. 4 pour un exemple avec 3 variables d'entrée. Les zones en bleu sont celles autour des points déjà échantillonnés, et les zones en jaune sont les régions d'intérêt. Une structure en réseau peut être observée, due au compromis entre exploitation et exploration.

Le prochain point proposé par l'algorithme sera celui qui maximise $S(\boldsymbol{x})$. Par exemple, le prochain point à tester pourrait être

$$\{x_1 = 0.35\ ,\ x_2 = 0.5\ ,\ x_3 = 0.15\}\,. \tag{9}$$

Alternativement, l'algorithme peut également proposer une région d'intérêt à l'expérimentateur. Cette région comprend tous les points ayant un score d'échantillonnage S supérieur à un seuil donné (par exemple 0.9). Par exemple, une région d'intérêt pourrait être :

$$\begin{cases} 0.3 \leq x_1 \leq 0.425 \\ 0.46 \leq x_2 \leq 0.535 \\ 0.11 \leq x_3 \leq 0.21. \end{cases} \tag{10}$$

puis l'expérimentateur choisira un point dans cette région.

Une explication peut également être donnée pour justifier le point/région proposé. Par exemple : "l'algorithme propose ce point/région à explorer à côté d'une zone déjà exploitée et qui a donné de bons résultats."

La proportion exploitation/exploration du prochain point testé peut également être fournie à l'expérimentateur :

$$p_{\text{exploitation}} = \frac{\hat{y}(\boldsymbol{x}_{next})}{\hat{y}(\boldsymbol{x}_{next}) + \lambda\, d(\boldsymbol{x}_{next})} \tag{11}$$

$$p_{\text{exploration}} = 1 - p_{\text{exploitation}}\,. \tag{12}$$

Enfin, des contraintes peuvent être prises en compte pour restreindre l'espace d'entrée ; par exemple $x_2 < 0.5$. Expérimentalement, ces contraintes pourraient être imposées par le dispositif expérimental, par exemple du fait des limites de la machine expérimentale ou du fait de lois physico-chimiques (comme la loi de miscibilité) dans le cas d'un mélange de matériaux.

5 Interprétabilité

Les avantages d'avoir des règles multidimensionnelles par rapport aux règles basées sur des ensembles flous 1D sont la possibilité de contrôler le nombre de règles souhaité (correspondant au nombre de clusters), d'éviter l'explosion du nombre de règles par rapport au nombre de dimensions, et de capturer interactions entre variables. Cependant, les règles multidimensionnelles sont moins interprétables que le cas unidimensionnel. Nous pouvons rendre le système multidimensionnel plus interprétable en établissant différentes catégories linguistiques par variable (par exemple faible/moyen/élevé) et en projetant les centres de chaque cluster sur chaque axe de variable [14].

Notre algorithme d'interprétabilité suit les étapes suivantes :

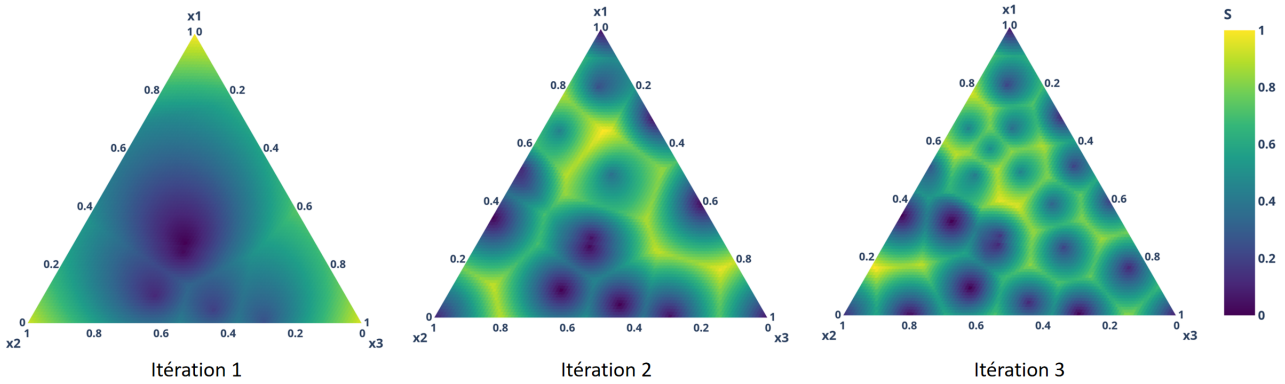

FIGURE 4 – Diagrammes ternaires illustrant l'évolution du score d'échantillonnage S à différentes itérations, pour un cas avec 3 variables d'entrée.

— Nous divisons chaque variable d'entrée et de sortie en différents ensembles flous triangulaires/trapézoïdaux f, dont les sommets correspondent aux centres des clusters projetés sur chaque axe. Si deux ensembles flous sont trop proches (par exemple distance $<$ distance seuil), nous les fusionnons ;

— Pour une variable x_i $(i \in [1; N+1])$, en notant m_{c_i} la i^{ieme} composante du centre $\boldsymbol{m}_c$, alors le sous-ensemble associé au cluster R est celui avec le degré d'appartenance le plus élevé $\mu^f(m_{c_i})$.

Un cluster sera associé à $N+1$ sous-ensembles flous (un par variable).

La figure 5 illustre le système de règles interprétables obtenu pour un cas avec 1 variable d'entrée x et 3 clusters. Les règles 1D sont utilisées comme substituts des règles multidimensionnelles pour aider l'utilisateur à comprendre ce modèle.

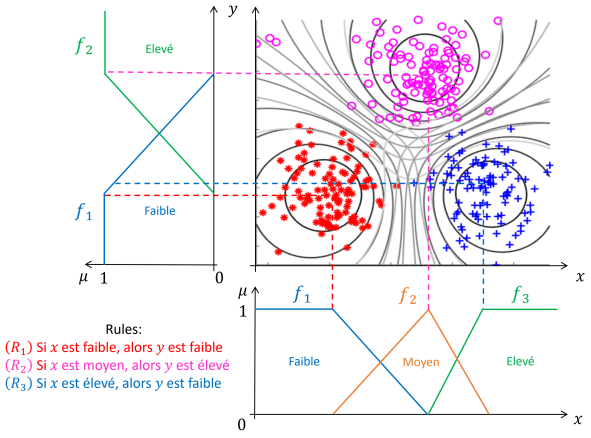

FIGURE 5 – Illustration de 3 clusters avec des degrés d'appartenance représentés par des lignes de niveau noires (tirées de [13]) et des ensembles flous f_i des variables x et y. L'association de chaque centre de cluster à ces ensembles flous 1D rend les règles plus interprétables.

6 Résultats expérimentaux

Dans cette section, nous présentons les résultats de différents tests pour caractériser notre algorithme. Par souci de reproductibilité, nous donnons d'abord quelques détails sur la mise en œuvre.

6.1 Considérations d'implémentation

Pour effectuer le clustering multidimensionnel, nous avons hybridé deux approches : nous utilisons la méthode de clustering hiérarchique [17] pour obtenir les centres des clusters et nous utilisons la fin de l'approche c-means floue [11] pour déterminer les fonctions d'appartenance. En effet, le clustering hiérarchique a l'avantage d'être totalement déterministe. Nous avons déterminé empiriquement le nombre de clusters pour chaque ensemble de données.

L'optimisation de la fonction de coût Eq. 7 peut être implémentée par la méthode Trust Region Reflective (TRF) [18], l'algorithme de Levenberg-Marquardt [9], l'algorithme des moindres carrés récursifs [8] ou descente de gradient [19, 10]. Nous avons utilisé l'algorithme TRF car il s'agit d'une méthode robuste (peu sensible au choix du point de départ), bien adaptée aux problèmes complexes avec des résidus non linéaires, adaptée aux grands problèmes clairsemés avec des bornes, et qui ne nécessite pas d'hyperparamètres supplémentaires.

6.2 Jeux de données jouet

Nous avons d'abord évalué notre méthode sur un jeu de données jouet que nous avons généré à partir d'une fonction sinus avec éventuellement plusieurs entrées. Un petit nombre d'entrées nous aide à visualiser les résultats pour les qualifier, tandis qu'un grand nombre permet de valider notre algorithme.

Fonction sinus à 1 variable d'entrée Nous avons d'abord testé notre algorithme de régression décrit dans la partie 3 avec le cas simple de la fonction $f(x) = \sin(2\pi x)$, avec 6 points initiaux et 2 clusters flous. Le résultat est tracé sur la Fig. 6, avec les valeurs de sortie prédites et les valeurs de sortie réelles.

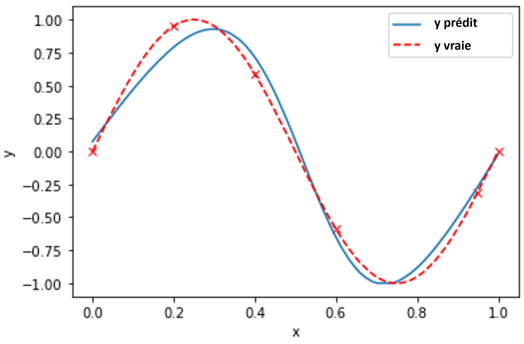

FIGURE 6 – Illustration de la sortie prédite $\hat{y}$ à l'aide de l'algorithme de régression basé sur le clustering flou.

Fonction sinus à 3 variables d'entrée Nous avons ensuite testé notre algorithme pour la fonction objectif sinus suivante avec 3 variables d'entrée :

$$f(\boldsymbol{x}) = \left| \prod_{k=1}^{3} \sin(k\pi x_k) \right| . \tag{13}$$

Cette fonction objectif a été choisie car elle présente une forme non triviale avec plusieurs maxima et un maximum global, avec des valeurs de sortie comprises entre 0 et 1, et parce qu'elle peut être tracée dans un diagramme ternaire (voir Fig. 7). Cela nous aide également à caractériser l'approche puisque nous n'aurons jamais un plan expérimental complet à partir d'une application réelle.

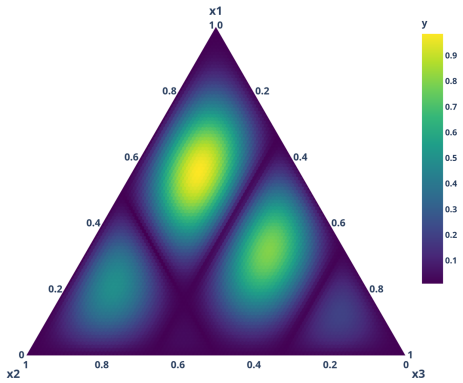

FIGURE 7 – Diagramme ternaire de la fonction objectif f donnée par l'équation (13).

Chaque axe x_i contient 100 valeurs de 0 à 1, donc au total nous avons 5151 points dans l'espace d'entrée. Notre objectif est de converger le plus rapidement possible vers la valeur optimale.
On choisit initialement 5 points aléatoires et on effectue 50 itérations (avec un point testé par itération). L'efficacité de notre algorithme est mesurée avec le critère du nombre d'itérations M nécessaires pour atteindre 80% de la valeur optimale de y (qui est de 0,984 dans notre cas d'étude). La figure 8 montre la meilleure valeur y obtenue parmi les points testés jusqu'à une itération donnée ; 80% de la valeur optimale est atteinte à l'itération $M = 24$.

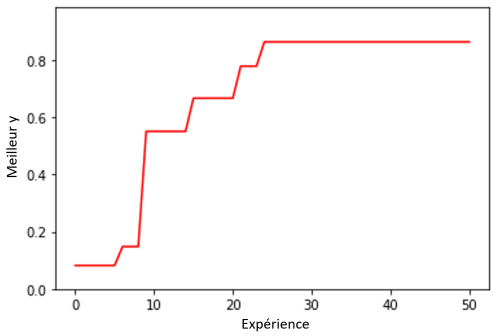

FIGURE 8 – Meilleure valeur de y obtenue pour chaque itération.

La proportion d'exploitation/exploration du point testé à chaque itération est tracée dans l'histogramme Fig. 9.

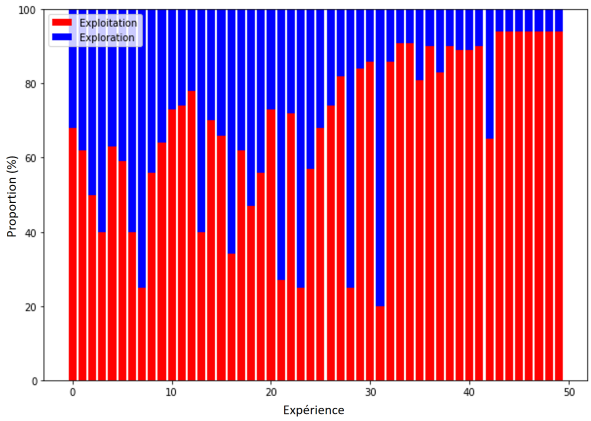

FIGURE 9 – Proportion exploitation/exploration pour chaque point testé.

On observe, comme prévu, que la proportion d'exploration diminue à mesure que le nombre de points échantillonnés augmente. A noter qu'il y a encore quelques explorations même après un nombre élevé d'expériences ; en effet, il vaut mieux continuer à explorer pour éviter de tomber sur un maximum local.
Nous avons ensuite testé la sensibilité de notre approche à son initialisation. En effet, le nombre d'expériences M nécessaires pour atteindre 80% de la valeur optimale dépend de la pertinence des points aléatoires initiaux. Nous avons répété la simulation 100 fois où à chaque simulation nous avons 5 points initiaux aléatoires avec $y < 0,1$ (c'est-à-dire que nous avons choisi des points non pertinents). Nous avons évalué que le nombre d'itérations nécessaires pour atteindre 80% de la valeur optimale est $M = 20 \pm 12.5$. Cette variabilité est due au fait que l'algorithme est sensible aux points initiaux, d'autant plus quand le nombre de points initiaux est faible.

Fonction sinus avec $N > 3$ variables d'entrée Pour nous rapprocher d'un problème du monde réel, nous avons testé notre approche avec plus de variables. Pour cela, nous considérons la fonction objectif suivante avec des $N \geq 3$

variables d'entrée :

$$f_N(\boldsymbol{x}) = \left| \prod_{k=1}^{N} \sin(i\pi x_k) \right| \tag{14}$$

avec $\boldsymbol{x} = (x_1, x_2..., x_N)$ variables continues, discrètes ou catégorielles. Les valeurs de y sont comprises entre 0 et 1. Plus précisément, nous étudions le cas de variables discrètes d'entrée $3 \leq N \leq 10$ (avec 5 valeurs différentes chacune). Ceci simule une expérience d'optimisation de composition dont les valeurs sont très contraintes.
Nous utilisons ce dernier jeu de données jouet pour comparer notre approche avec l'optimisation bayésienne (OB). L'OB diffère de notre algorithme flou pour les étapes suivantes [1, 20] :

— Pour l'OB, nous évaluons la fonction de substitution à l'aide du processus gaussien (GP) ou de l'algorithme de Parzen structuré arborescent (TPE), pour modéliser la fonction objectif avec une valeur moyenne m et une dispersion σ. Cet algorithme rend l'OB non déterministe. Pour notre algorithme flou, la fonction surrogate est obtenue à partir de l'algorithme de régression basé sur le clustering flou décrit en partie 3.

— Pour l'OB, le compromis entre exploitation et exploration est modélisé en utilisant la fonction d'acquisition "Expected Improvement" : $EI(\boldsymbol{x}) = (m(\boldsymbol{x}) - f(\boldsymbol{x}^*))\phi\left(\frac{m(\boldsymbol{x})-f(\boldsymbol{x}^*)}{\sigma(\boldsymbol{x})}\right) + \sigma\Phi\left(\frac{m(\boldsymbol{x})-f(\boldsymbol{x}^*)}{\sigma(\boldsymbol{x})}\right)$, avec ϕ/Φ la densité de probabilité/fonction de partition de la distribution normale et $f(\boldsymbol{x}^*)$ la meilleure valeur y obtenue jusqu'à présent. Pour notre algorithme flou, ce compromis exploitation/exploration est modélisé à travers le score d'échantillonnage $S(\boldsymbol{x}) = \hat{y}(\boldsymbol{x}) + \lambda\, d(\boldsymbol{x})$.

De manière analogue à notre algorithme, l'OB est répétée sur un certain nombre d'itérations. Chaque boucle fournit des informations supplémentaires jusqu'à atteindre une valeur optimale. L'algorithme TPE est efficace en termes de calculs et bien adapté aux problèmes d'optimisation de grande dimension avec une fonction objectif coûteuse [21]. De plus, il est bien adapté aux problèmes d'optimisation impliquant des variables discrètes et catégorielles, ainsi que des variables continues [22].
Pour les deux algorithmes, nous évaluons le nombre d'itérations M nécessaires pour atteindre 80% de la valeur optimale pour un nombre N donné de variables d'entrée et pour 5 points initiaux donnés (voir Fig. 10). Dans le cas de l'OB, en raison du caractère aléatoire du calcul de la fonction surrogate, nous avons dû répéter la simulation plusieurs fois pour obtenir une valeur moyenne. Nous observons que globalement notre algorithme flou donne un meilleur résultat que l'OB, il converge avec moins d'itérations :

$$\forall N, M_{fuzzy} < M_{BO}. \tag{15}$$

De plus, l'OB est une méthode non déterministe et on observe que la disparité de convergence est assez importante (voir les barres d'écart type élevées en bleu sur la figure).

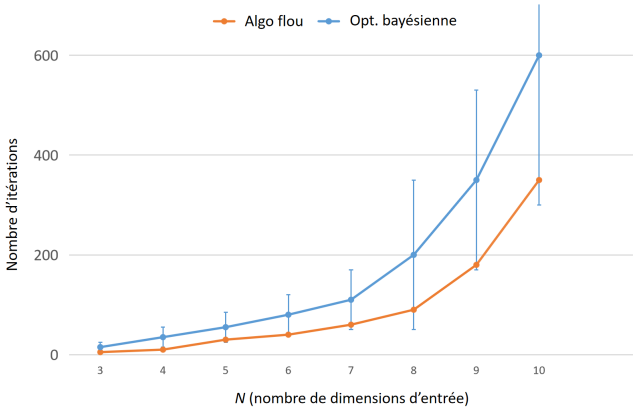

FIGURE 10 – Nombre d'itérations nécessaires pour atteindre 80% de la valeur optimale - Cas avec N variables discrètes.

6.3 Jeu de données réel

Nous avons ensuite testé notre approche sur un ensemble de données réelles du UC Irvine Machine Learning Repository appelé "Concrete Compressive Strength" [1].
Il comprend 8 variables d'entrée et le but est de maximiser la variable de sortie normalisée "Concrete compressive strength (MPa)". Cet ensemble de données comprend 1030 instances. En utilisant un processus de validation croisée (avec 80% de l'ensemble d'entraînement et 20% de l'ensemble de tests), nos simulations sur cet ensemble de données ont montré que la fonction de substitution de notre algorithme flou conduit à une meilleure prédiction de régression que le processus gaussien d'optimisation bayésienne : $RMSE = 8,7 \pm 2,9$ pour notre algorithme flou, et $RMSE = 15,1 \pm 3,6$ pour le processus gaussien. En utilisant notre substitut décrit dans la partie 5, nous obtenons $RMSE = 10.7 \pm 4.9$; ce résultat est moins bon que notre algorithme, soulignant la nécessité d'utiliser les règles N-dimensionnelles décrites dans la partie 3.2.
Pour déterminer le nombre d'expériences pour atteindre un point optimal, nous avons effectué le processus suivant : nous avons choisi 5 points aléatoires parmi les 1030 instances avec une mauvaise valeur de sortie $y < 0,1$; une expérience consiste ici à prendre un point parmi les instances choisies par l'algorithme et on souhaite converger rapidement vers un point optimal. La simulation complète est répétée plusieurs fois pour obtenir une valeur moyenne et un écart type. Le nombre d'expériences nécessaires pour atteindre 80% de la valeur optimale de y est $M = 9 \pm 6,7$.
Qualitativement, notre algorithme est particulièrement utile lorsqu'il s'agit de données expérimentales. En effet, l'explicabilité est nécessaire pour comprendre pourquoi un point/une région précis est proposé par l'algorithme. Le déterminisme est également très important puisque l'expérimentateur ne souhaite pas se voir proposer un point différent à chaque fois qu'il exécute l'algorithme. De plus, des contraintes peuvent être prises en compte sur la base

1. https://archive.ics.uci.edu/dataset/165/concrete+compressive+strengt

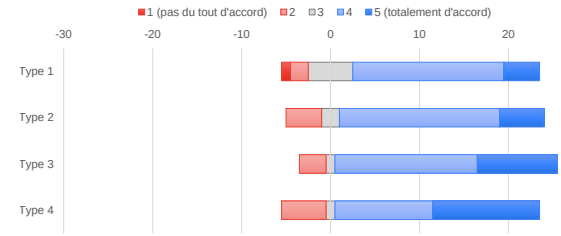

FIGURE 11 – Réponses à la question “Êtes-vous satisfait de la prochaine expérience proposée par l'algorithme ?”

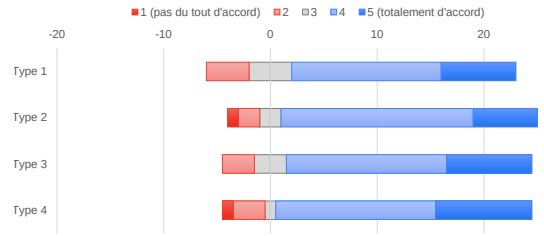

FIGURE 12 – Réponses à la question “Faites-vous confiance au choix de l'algorithme ?”

de connaissances expérimentales et théoriques.

7 Considération de l'utilisateur final

Nous avons évalué l'interaction avec les utilisateurs finaux à l'aide d'un questionnaire (évaluation basée sur l'humain). Le panel est constitué de 29 personnes travaillant dans le domaine de la science des matériaux, allant des chercheurs académiques aux chercheurs industriels, âgées de 22 à 62 ans. Pour recueillir leurs avis, nous avons utilisé une échelle de Likert en 5 points, allant de totalement en désaccord à totalement d'accord.

Le questionnaire décrit une situation basée sur un mélange de 3 composés (x_1, x_2, x_3) et une propriété (y) qui va de 0 à 1. Sur un diagramme ternaire, 17 expériences précédentes sont représentées avec les valeurs respectives de la propriété. Nous avons demandé au panel de comparer 4 résultats différents :

- Type 1 : l'algorithme donne exactement les valeurs suivantes pour x_1, x_2, x_3, comme dans l'équation. 9 ;
- Type 2 : idem Type 1 avec une explication (ex : “exploiter une zone déjà explorée et qui a donné de bons résultats”) ;
- Type 3 : l'algorithme donne une région d'intérêt comme dans l'équation 10 ;
- Type 4 : identique au Type 3 et avec la même explication que dans le Type 2.

La figure 11 montre les réponses à la question : “êtes-vous satisfait de la ou des prochaines expériences proposées par l'algorithme ?”. Les résultats sont majoritairement positifs, mais le troisième type de production semble avoir les meilleures notes (c'est-à-dire une région sans explication). De plus, la Figure 12 montre les réponses à la question : “faites-vous confiance au choix de l'algorithme ?”. Les deux types qui n'apportent aucune explication ont des résultats moins positifs. Cependant, les deux types qui fournissent une explication aux utilisateurs ont des résultats plus positifs, mais ils ont également un panéliste qui est totalement en désaccord. Nous avons regardé de plus près ses réponses et elles sont parfois contradictoires : cela peut être considéré comme une valeur aberrante.

Nous avons demandé aux panélistes quelle est leur sortie préférée : 12 d'entre eux répondent au Type 4, 8 répondent au Type 2, 5 pour le type 3 et enfin 4 pour le Type 1. Ainsi, les deux sorties préférées sont accompagnées d'explications.

Il est important de mentionner que nous avons demandé aux panélistes s'ils avaient peur de l'intelligence artificielle dans leur travail : 7 panélistes sont tout à fait d'accord et 10 d'entre eux sont d'accord. 9 d'entre eux sont restés neutres, ce qui signifie que seuls 3 d'entre eux n'ont pas peur.

Les résultats confirment ce que nous attendions : les utilisateurs finaux préfèrent avoir un choix et une explication, ce qui signifie que la méthode finale doit fournir la région de la prochaine expérience et une explication de la recommandation. Dans un commentaire, un panéliste a écrit qu'il préférait avoir plusieurs expériences possibles plutôt qu'une seule, mais puisque l'algorithme a choisi une expérience centrale, son choix serait le même. Il souligne l'importance de prendre en compte les préférences humaines dans de tels outils pour accroître leur acceptabilité.

8 Conclusion et perspectives

En conclusion, nous résumons les avantages de notre approche en fonction des résultats obtenus. Notre algorithme est transparent en conséquence directe du caractère interprétable de ses paramètres, de la prédominance d'un cluster dans chaque région de l'espace d'entrée-sortie, de la simplicité et de la nature linguistique de ses règles floues ; il est déterministe, c'est-à-dire qu'il converge vers les mêmes valeurs à chaque exécution ; il est nettement plus rapide que les approches méta-heuristiques telles que les algorithmes évolutionnaires ; il est robuste au surentraînement et résilient au bruit grâce à la contribution fusionnée des clusters ; enfin, des contraintes issues de la littérature peuvent être appliquées pour réduire l'espace de recherche d'entrée. Cette approche a le mérite d'être interprétable, intuitive, et peut être d'une réelle aide aux expérimentateurs.

L'originalité de notre algorithme inclut la transformation de clusters flous multidimensionnels en ensembles flous 1D interprétables, et la définition d'une fonction de score d'acquisition/échantillonnage à partir des variables $\hat{y}$ (valeur de sortie prédite) et d (distance à le point échantillonné le plus proche).

Les possibilités d'amélioration incluent la vitesse de calcul dans le cas de grande dimension, une meilleure détection des extrema locaux et une optimisation multi-objectifs.

Remerciements

Ce travail est financé par le Programme Transversal de Compétences en Matériaux et Procédés du CEA.

Références

[1] S. GREENHILL et al. “Bayesian optimization for adaptive experimental design : A review”. In : *IEEE access* 8 (2020), p. 13937-13948.

[2] D.S. BEM et al. “Combinatorial experimental design using the optimal-coverage algorithm”. In : *Experimental Design for Combinatorial and High Throughput Materials Development* (2003).

[3] S. TAKAMOTO et al. “Towards universal neural network potential for material discovery applicable to arbitrary combination of 45 elements”. In : *Nature Communications* 13.2991 (2022).

[4] A. PAGLIARO et P. SANGIORGI. “AI in Experiments : Present Status and Future Prospects”. In : *Applied Sciences* 13.10415 (2023).

[5] B. CAO et al. “How To Optimize Materials and Devices via Design of Experiments and Machine Learning : Demonstration Using Organic Photovoltaics”. In : *ACS Nano* 12.8 (2018), p. 7434-7444.

[6] F. REN et al. “Accelerated discovery of metallic glasses through iteration of machine learning and high-throughput experiments”. In : *Science Advances* 4.4 (2018), eaaq1566.

[7] D. XUE et al. “Accelerated search for materials with targeted properties by adaptive design”. In : *Nature communications* 7.1 (2016), p. 1-9.

[8] S.L. CHIU. “Fuzzy model identification based on cluster estimation”. In : *Journal of Intelligent & fuzzy systems* 2.3 (1994), p. 267-278.

[9] K. WIKTOROWICZ. “RFIS : regression-based fuzzy inference system”. In : *Neural Computing and Applications* 34 (juill. 2022).

[10] J. VIAÑA et al. “Explainable fuzzy cluster-based regression algorithm with gradient descent learning”. In : *Complex Engineering Systems* (2022).

[11] J.C. BEZDEK, R. EHRLICH et W. FULL. “FCM : The fuzzy c-means clustering algorithm”. In : *Computers & Geosciences* 10.2 (1984), p. 191-203. ISSN : 0098-3004.

[12] I.B. TÜRKŞEN. “A review of developments from fuzzy rule bases to fuzzy functions”. In : *2012 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)*. 2012, p. 1-5.

[13] M.-J.e LESOT, M. RIFQI et B. BOUCHON-MEUNIER. “Fuzzy Prototypes : From a Cognitive View to a Machine Learning Principle”. In : *Fuzzy Sets and Their Extensions : Representation, Aggregation and Models*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, p. 431-452. ISBN : 978-3-540-73723-0.

[14] G. TSEKOURAS et al. “A hierarchical fuzzy-clustering approach to fuzzy modeling”. In : *Fuzzy sets and systems* 150.2 (2005), p. 245-266.

[15] S. GUILLAUME et B. CHARNOMORDIC. “Generating an interpretable family of fuzzy partitions from data”. In : *IEEE Transactions on Fuzzy Systems* 12.3 (2004), p. 324-335.

[16] S. MANEEWONGVATANA et D.M. MOUNT. “Data Structures, Near Neighbor Searches, and Methodology”. In : American Mathematical Society, 2002. Chap. Analysis of approximate nearest neighbor searching with clustered point sets.

[17] L.-J. LI et Y.-L. LIANG. “A Hierarchical Fuzzy Clustering Algorithm”. In : *International Conference on Computer Application and System Modeling* (2010).

[18] A.R. CONN, N.I.M. GOULD et P.L. TOINT. “Trust-Region Methods”. In : *MPS-SIAM Series on Optimization 1. SIAM and MPS, Philadelphia* (2000).

[19] G.E. TSEKOURAS et al. “A fuzzy clustering-based algorithm for fuzzy modeling.” In : *WSEAS Transactions on Systems* 3.5 (2004), p. 1958-1963.

[20] S. WATANABE. “Tree-structured Parzen estimator : Understanding its algorithm components and their roles for better empirical performance”. In : *arXiv preprint arXiv :2304.11127* (2023).

[21] J. BERGSTRA, D. YAMINS et D.D. COX. “Making a Science of Model Search : Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures”. In : *Proc. of the 30th International Conference on Machine Learning* (2013).

[22] T. AKIBA et al. “Optuna : A next-generation hyperparameter optimization framework”. In : *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, p. 2623-2631.

Index des notations

- N nombre de dimensions d’entrée
- $\boldsymbol{x}_s$ points déjà échantillonnés, $\boldsymbol{x}$ point de l’espace d’entrée, $\boldsymbol{x}_{next}$ prochain point testé
- f ensemble flou
- c cluster, C ensemble des clusters
- $\boldsymbol{m}_c$ centre du cluster c, n_c nombre de clusters, μ_c degré d’appartenance au cluster c
- R règle floue
- $\boldsymbol{\alpha}_c, \beta_c$ coefficients de régression relatifs au cluster c
- $\hat{y}_c$ sortie prévue pour une entrée $\boldsymbol{x}$, par rapport au cluster c
- $\hat{y}$ sortie globale prédite pour une entrée $\boldsymbol{x}$
- M nombre d’itérations nécessaires pour atteindre un point optimal