Contents

Chapter 1

Introduction

1.1 Background and Motivation

1.2 Problem Statement

1.3 Contributions

3.1	Intrinsic XAI through Self-Organization Analysis: Developed the oCELL algorithm and formalized a novel methodology leveraging agent self-organization structures to perform model introspection and infer underlying function properties for enhanced explainability and interpretability. [5]
3.1	Empirical Evaluation of oCELL Performance: A low-dimensional comparative study demonstrating oCELL's competitive performance against SotA ensemble algorithms. [5]
3.2	Non-Stationary Adaptation Capabilities: Development of the kCELL algorithm and a study demonstrating its ability to adapt to non-stationary dynamics for forward prediction, using introspection to analyze the system's behavior.
4	Integration for Constrained Optimal Control: Formalization of a novel coupling method to integrate the kCELL learning mechanism into traditional optimization solvers for solving complex constrained optimal control problems with learned models.
4	Introspection-Enhanced Safe Control: Proposal of a novel methodology that utilizes kCELL's intrinsic introspection capabilities to enhance the safety and performance of the solved constrained optimal control problems with learned models.
4	Constraint-Impact Explainability Method: Proposal of a novel XAI method leveraging kCELL's structure to quantify and explain the impact of constraints on optimization outcomes in safe control.
5	Scalable CELL Spatialization: Proposal of Principal Component Analysis (PCA) as a novel technique to scale the CELL paradigm to higher-dimensional feature spaces, including a comparative evaluation.
5	Scalable Gaussian Process Learning (SGP-CELL): Development of the SGP-CELL algorithm, a novel machine learning model that leverages the CELL paradigm to achieve online learning and scalability in Gaussian Processes (GP) with respect to the number of data points.

Table 1.1: Summary of Contributions

Chapter 2

State of the Art

This chapter provides a comprehensive review of the literature on speed recommendation systems. We decompose the problem into two sub-problems: a *modeling problem*, which involves continuously estimating vehicle dynamics under driver-specific influence to enable personalized and realistic speed recommendations; and a *control problem*, which leverages the learned model to optimize speed profiles with respect to predefined objectives and operational constraints. To establish a rigorous foundation and position our approach within the field, we examine state-of-the-art methods across three relevant domains: speed control and recommendation, online machine learning, and optimal control.

2.1 Speed Recommendation and Speed Control

2.2 Online Machine Learning

We provide an overview of related works about multiagent learning, ensemble methods and eXplainable AI (XAI) to better situate the CELL paradigm within the existing literature.

2.2.1 Multi-Agent Learning

Multi-agent systems (MAS) have gained popularity due to their ability to solve complex problems by breaking them down into simpler sub-problems that can be easily addressed by autonomous agents, whether interconnected or not [18]. The interaction rules between agents or with the environment are defined by the system designer to achieve a specific objective. The use of MAS has proven effective in fields such as civil engineering [47] or electrical engineering, particularly with issues related to smart grids [43]. In more recent years, reinforcement learning has been seriously considered to bypass the phase of designing rules that define agent behavior. Indeed, in Multi-Agent Reinforcement Learning (MARL), agents' behaviors are learned using reinforcement signals obtained by continuously interacting with an environment [12].

In this chapter, we present various two instantiations of CELL, a MAS paradigm derived from the Self Adaptive Context Learning (SACL) [6] that combines both approaches to tackle online supervised learning problems. Agents of the system update using both reinforcement

learning signals and cooperation rules. Unlike MARL, CELL requires fewer environment interactions and focuses on specialized agents collaborating within a supervised learning framework, differing from MARL's dynamic interactions aiming to maximize cumulative rewards through adaptive strategies (*competitive* or *cooperative*) [12].

2.2.2 Ensemble Learning

To achieve more accurate predictions and provide better approximations of nonlinear functions, it is common to aggregate multiple models for making predictions.

Ensemble learning is based on the emergence of collective intelligence within a set of weak learners. A weak learner is a model whose performance is at least as good as a model making random predictions. During the learning process, a set of models (which may differ from one another) are trained in parallel or sequentially [40]. The objective is to encourage diversity among the models so that they do not all capture the same patterns in the data [17]. An input is transmitted to all weak learners, each of which makes a prediction proposal. A heuristic is implemented to select one of the proposals or to weight each of them in order to construct the final prediction. We distinguish several major approaches in ensemble learning which are Boosting, Bagging and Stacking.

Figure 2.1: Visual comparison of bagging, boosting and stacking

In Bagging [9] multiple models (usually homogeneous) are trained in parallel on different subsets of the training data to introduce diversity, reduce variance and improve stability. Then the final prediction is obtained by averaging the predictions of the weak learners (regression) or by majority voting (classification). Unlike Bagging, in Boosting [21], the models are trained sequentially. Each new model focuses on correcting the errors of the previous ones to reduce bias and improve accuracy. Then the final prediction is obtained from a weighted average of the predictions of the all the weak learners. Finally, there is Stacking [54] that falls under meta-learning. We train in parallel a set of heterogeneous models and then a meta-model is trained to combine the predictions of each model to obtain the final output of the system, in order to capture complementary strengths of each learning algorithms involved in the learning process.

The family of systems presented in this chapter, CELL, falls within ensemble learning and we could label it as a Bagging approach because we consider a set of weak learners as a collection of self-organizing cooperative agents. Each one of them is a local expert on the function to be approximated.

2.2.3 XAI

The field of eXplainable Artificial Intelligence (XAI) addresses the opacity of complex models by providing insights into their decision-making processes. However, interpretability and explainability are defined in various ways [23, 19] and often used interchangeably. Thus, following the distinction made in [11], we differentiate between interpretability and explainability. **Interpretability** is defined as the inherent ability to understand how the model

works as a whole, focusing on the model's structure and mechanisms (i.e transparency). In contrast, **Explainability** is associated with the methodologies used to communicate the reasoning for a specific decision taken by a model.

Approaches such as deep learning achieve good performance across a wide variety of tasks. However, these approaches generate highly complex models. There must be a compromise between the model's performance and its ability to produce explainable predictions and interpretable structures [32]. Some approaches were developed to adress the explainability of predictions. SHAP looks for the impact of each feature on the prediction using cooperative game theory [35] and LIME builds an interpretable surrogate model that approximate a black-box model locally [42].

Within this discourse, machine learning models are typically categorized as white-box or black-box models [34]. **Black-box** models have opaque and complex inner workings that are difficult for an external observer to understand. This categorisation includes deep learning models [31] and some ensemble models [10, 40, 14, 29]. Conversely, **White-box** models have simpler and less opaque inner workings. These models, considered more explainable than black-box models, include linear regression models [51], decision trees [44] , and other rule-based approaches. To achieve explainability, white-box models are preferred.

The explainability of a prediction is partly linked to the concept of uncertainty. An informed decision-making process must take uncertainty into account. We distinguish between epistemic uncertainty and aleatoric uncertainty [15, 25]. **Aleatoric uncertainty** arises from the inherent natural variations in the studied phenomena. It may be due to random fluctuations, measurement errors, or other unpredictable factors. **Epistemic uncertainty** on the other hands, stems from a lack of knowledge or complete understanding of the studied phenomenon. This type of uncertainty is related to the lack of data in certain regions of the feature space.

2.3 Optimal Control

2.4 Positioning

Chapter 3

Context Ensemble Local Learning (CELL)

To achieve speed recommendation, the behavior of a vehicle driven by a human driver in response to speed recommendation set points need to be modeled. Therefore, we need a modeling algorithm able to perform online learning and learn efficiently seeing a data point once while having transparent inner workings and explainable predictions. For that reason we explored the use of multiagent systems as a mean of solving supervised learning tasks.

As showed in [6, 20], a supervised learning problem can be modeled as a multiagent system. This perspective offers several advantages, including design simplicity, transparency, interpretability and explainability properties that naturally emerge from the structural organization of agents.

Building upon the Self Adaptive Context Learning (SACL) paradigm [6], this chapter introduces the CELL (Context Ensemble Local Learning) learning paradigm, which provides the core hypotheses and theoretical grounds for designing spatialized multiagent learning systems suited for modeling the dynamics of complex systems from online streaming data. This paradigm addresses online supervised learning through self-organization of multiple local expert agents paving the feature space. These agents are created and updated dynamically according to predefined learning rules. Each agent occupies a specific region in feature space, representing the area where it is most confident in the quality of its predictions. Contrary to previous works on this type of system [6, 20], we introduce novelties regarding explainability and interpretability while exploring new cooperation mechanisms between agents and new spatialization approaches.

By leveraging the spatialization of local experts and the inherent transparency of the model, we derive unique informative explainability properties that provide valuable insights about the approximated function. Furthermore, we outline practical guidelines for scaling with the number of agents, keeping a bounded computational complexity. Therefore, the CELL paradigm is introduced through two distinct learning systems: oCELL and kCELL.

First, in section 3.1, oCELL is introduced, its capabilities are presented in terms of predictive performance and explainability, as well as its structural limitations. Section 3.2 then describes kCELL, a more robust and expressive instantiation of CELL that addresses several of oCELL's limitations while preserving its intrinsic explainability and interpretability properties. Finally, Section 4.1 discusses practical considerations for efficient implementation,

including strategies for obtaining differentiable operations, GPU parallelization, and spatial indexing to optimize learning speed and inference.

3.1 oCELL: Multiagent Ensemble Learning with Orthotopes

This section introduces the first instantiation of CELL, referred to as oCELL (orthotope CELL). CELL systems are multiagent systems composed of autonomous agents, each possessing local knowledge and specific capabilities. These systems build upon the Self-Adaptive Context Learning (SACL) paradigm [6] to address online supervised learning tasks.

The objective is to approximate a function $f : \mathbb{R}^n \to \mathbb{R}^m$ that maps feature vectors to target vectors from a continuous stream of data. oCELL processes incoming data points by routing them to its primary components, the context agents. These agents are dynamically created and updated according to predefined learning rules. Each agent occupies a distinct region of the feature space defined as an orthotope, where it maintains the highest confidence in its predictions. This spatial organization allows agents to determine *when* they should contribute to the prediction process. To determine *what* to predict, each agent maintains a local machine learning model over its confidence region.

First, in 3.1.1, the internal structure of context agents is described, including their spatialization in the feature space and their prediction mechanisms. Then, Section 3.1.2 introduces the learning rules that cover agent creation, adaptation and self-organization leading to the emergence of learning in the system. Section 3.1.3 presents a comparative study evaluating the performance of oCELL against other machine learning algorithms on a two-dimensional benchmark. Subsequently, section 3.1.4 presents an analysis of the capabilities of oCELL in terms of interpretability and explainability, which naturally arise from the spatial organization of agents. Finally, Section 3.1.5 discusses the limitations of oCELL and outline directions for future research, some of which are addressed in 3.2.

3.1.1 Context Agents

The main entities of oCELL are context agents. A context agent, denoted as

$$\mathcal{A}_i = \{\phi_i, f_i\}$$

is defined by two core components: an activation function $\phi_i(x)$ and a prediction function $f_i(x)$. The activation function determines whether the agent should contribute to the prediction for a given input x, while the prediction function provides the corresponding output.

Each agent can adapt the parameters of its activation and prediction functions based on reinforcement signals derived from its performance, enabling it to refine its behavior according to local conditions in feature space. Conceptually, a context agent acts as a local expert for the target function with activation function governing *when* the agent predicts, and the prediction function specifying *what* it predicts.

First, the spatialization mechanism of context agents based on orthotopes is described. Then the prediction mechanisms arising from interactions within an agent's neighborhood are detailed.

Spatialization with Orthotopes

To ensure transparency and interpretability, oCELL spatialize context agents using orthotopes (also referred to as hyper-rectangles in the literature). Orthotopes provide a clear geometric representation of an agent's confidence region in the feature space, allowing precise evaluation of local structures and facilitating shape adaptation during self-organization. This representation allows independent manipulation of each feature dimension, simplifying expansion and retraction operations.

Learning with orthotopes has been explored in previous works on supervised learning [20, 30]. These approaches typically partition the feature space into orthotopes and model the target function locally. For instance, [30], employs a gradient boosting to learn orthotope bounds, where each region corresponds to a simple constant model. However this approach is not suitable for online learning from data streams.

oCELL borrows a similar spatialization principle to [20], which uses the SACL paradigm [6] to progressively pave the feature space with context agents to address classification tasks. However, unlike [20], oCELL addresses issues specifically related to regression tasks such as the need for a smoother spatialization of agents to smooth prediction accuracy (i.e smooth the learned function).

Each agent's activation function defines a p-dimensional orthotope in the feature space. For each feature dimension $j \in \{1, \ldots, n\}$, the orthotope is parameterized by a lower bound l_j and an upper bound h_j such as

$$\mathcal{H}_i = [l_1, h_1] \times \cdots \times [l_n, h_n] \tag{3.1}$$

The volume $v(\mathcal{A}_i)$ of the orthotope associated to $\mathcal{A}_i$ is defined by

$$v(\mathcal{A}_i) = \prod_{j=1}^{n} (h_j - l_j) \tag{3.2}$$

An observation $x \in \mathbb{R}^n$ is considered as intersecting an orthotope if, for all feature j,

$$l_j \leq x_j \leq h_j \tag{3.3}$$

Additionaly, an orthotope $\mathcal{H}_1$ intersects another orthotope $\mathcal{H}_2$ if, for all feature j,

$$\max(l_{1,j}, l_{2,j}) \leq \min(h_{1,j}, h_{2,j}) \tag{3.4}$$

with $l_{1,j}$,$l_{2,j}$ and $h_{1,j}$,$h_{2,j}$ denote the lower and upper bounds of $\mathcal{H}_1$ and $\mathcal{H}_2$.

Agents in oCELL are constructed based on the assumption of uniform knowledge over their confidence region, i.e the area delimited by the associated orthotope. We distinguish between activation and neighborhood. When an agent is activated by a point, it means it is confident in its expertise to predict for that point. When an agent is a neighbor of a point, it means it has doubts about its expertise to predict for that point. In other words, it is an agent that is a candidate to become an activated agent for that point. The way an agent updates itself differs depending on whether the agent is activated or a neighbor.

Definition 1. A Context Agent $\mathcal{A}$ is considered activated by an observation x if x intersects with the orthotope $\mathcal{H}$ associated with the activation function $\phi_{\mathcal{H}} : \mathbb{R}^n \mapsto \{0, 1\}$ of $\mathcal{A}$. In other words, we define the activation function of $\mathcal{A}$ as

$$\phi_{\mathcal{H}}(x) = \begin{cases} 1 & \text{if } \forall j,\ l_j \leq x_j \leq h_j \\ 0 & \text{else} \end{cases} \tag{3.5}$$

Definition 2. The neighborhood of an observation x is defined as an orthotope $\mathcal{H}_{\text{neighborhood}}$ centered on x. A context Agent $\mathcal{A}_i$ is considered a neighbor of x if the orthotope $\mathcal{H}_i$ associated with the activation function ϕ_i of $\mathcal{A}_i$, intersects with $\mathcal{H}_{\text{neighborhood}}$(3.4).

For self-organization, agents must adapt their shape and position in the feature space. Unlike [20], which minimizes overlap to avoid ambiguity in classification (for example agents could push each other), oCELL allows overlapping regions to allow prediction smoothing through ensemble averaging. This property is particularly useful for regression tasks as it can be more informative to average the predictions of several weak models to smooth the learned function. It also facilitates integration into non-linear optimization processes (cf. section (**??**)). Therefore, an agent can change its shape by contracting or expanding its orthotope without worrying about the positions of other agents.

Definition 3. If a context agent $\mathcal{A}$ expands (resp. contracts) by a factor α in the direction of an observation $x \in \mathbb{R}^n$ at time t, then the upper bounds h_j^t and lower bounds l_j^t of the associated orthotope are updated according to the following relationship:

$$h_j^{t+1} = \begin{cases} \left(h_j^t - l_j^t\right) \varepsilon^{\frac{1}{k}} + l_j^t & \text{if } l_j^t \leq x_j \leq h_j^t \\ h_j^t & \text{else} \end{cases} \tag{3.6}$$

$$l_j^{t+1} = \begin{cases} \left(l_j^t - h_j^t\right) \varepsilon^{\frac{1}{k}} + h_j^t & \text{if } l_j^t \leq x_j \leq h_j^t \\ l_j^t & \text{else} \end{cases} \tag{3.7}$$

with

$$\varepsilon = \begin{cases} (1 + \alpha) & \text{if expansion} \\ (1 - \alpha) & \text{if retraction} \end{cases} \tag{3.8}$$

and k the number of features such that $l_j^t \leq x_j \leq h_j^t$. Thus, the new volume of the orthotope associated with $\mathcal{A}$ is given by the relation:

$$v_{\mathcal{A}}^{t+1} = \varepsilon v_{\mathcal{A}}^t \tag{3.9}$$

In summary, agents are spatialized through activation functions defining orthotopes that represent their *confidence region* (i.e area of expertise) in feature space. Each agent can adapt its bounds, expanding or contracting its associated orthotope as needed. Intuitively, larger agents act as *generalists*, while smaller agents serve as *specialists*. Consequently, complex regions of feature space tend to host many small agents, whereas simpler regions tend to be covered by fewer larger agents. This assumption aligns with the use of simple local models such as linear regressions, which we leverage in subsequent experiments to demonstrate the transparency and interpretability of oCELL.

Neighborhood Prediction

The objective of oCELL is to perform online regression from a continuous data stream. Spcifically, it aims at modeling the nonlinear dynamics of a complex system for subsequent use in an optimization pipeline for optimal control (see Chapter 4). Therefore, it is desirable to minimize gradient noise and ensure the learned function is smooth, as irregularities can significantly disrupt the optimization process [38].

To achieve this, oCELL allows multiple agents to contribute to the final prediction, in contrast to previous works [6, 16, 20], where a single agent was ultimately selected for prediction.

To obtain the prediction $\hat{y}$ from an observation x, the most competent agents are selected to predict the value of $\hat{y}$. If some agents are neighbors of x, then they each make a prediction proposal. If x has no neighbor, the k-closest agents are selected instead. This fallback mechanism prevents prediction failures in regions where the system has limited knowledge (i.e poor paving of the area around x). The final prediction is then given by the arithmetic mean of the proposals of selected agents such as

$$\hat{y} = \frac{1}{|D_{\text{selected}}|} \times \sum_{i \in D_{\text{selected}}} f_i(x)$$

where $D_{\text{selected}}$is the set of selected agents (neighbors or closest) and $f_i \in \mathbb{R}^m$ the internal prediction function of the i-th agent.

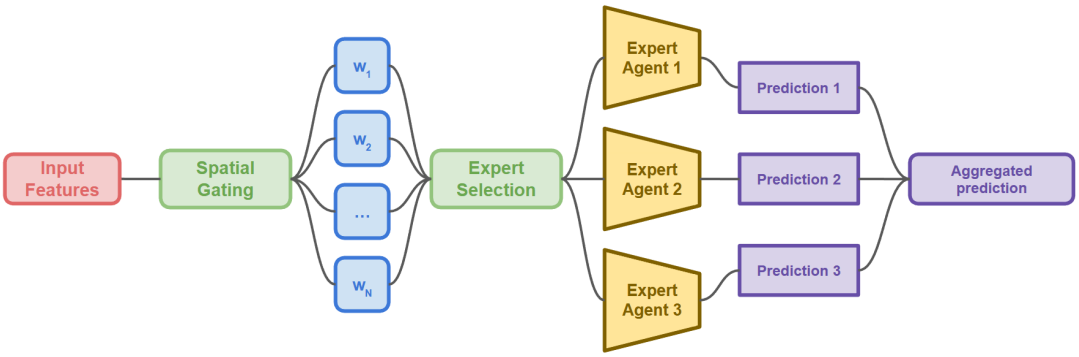


Figure 3.1: Illustration of the prediction process with agents in CELL paradigm

3.1.2 Learning Rules

In CELL systems, the design of learning rules revolves around a set of actions and trigger conditions that decide how to digest the continuous stream of (x_{new}, y_{new}) data. Some actions are tied to individual agents such as updating their model (prediction function) or shape (activation function); and others are more meta-level actions such as destroying or creating new agents. These actions are triggered by conditions. In oCELL, those conditions depend on the number of current neighbors, on the number of activated agents and on a feedback values calculated from the proposals of selected agents.

To determine the appropriate action for an agent, we define the prediction quality $L_{\mathcal{A}_i}$ of agent $\mathcal{A}_i$ as

$$L_{\mathcal{A}_i} = g\left(\hat{y}, y\right)$$

where $g : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$ is a distance measure between the proposal $\hat{y}$ and the true value y. The largest $L_{\mathcal{A}_i}$, the worse the quality of the proposal. In oCELL, we use the squared error as the g function but other types of error functions could be used instead.

We define two thresholds τ_{good} and τ_{bad} to partition the values of $L_{\mathcal{A}_i}$ into three regimes: $L_{\mathcal{A}_i} \in [0, \tau_{\text{good}}]$ for *good* predictions, $L_{\mathcal{A}_i} \in [\tau_{\text{good}}, \tau_{\text{bad}}]$ for *inaccurate* predictions, $L_{\mathcal{A}_i} \in [\tau_{\text{bad}}, +\infty[$ for *bad* predictions. These thresholds control the degree of accuracy expected from the system and influence the frequency of some rule activation.

Inaccuracy When there are $N_A \geq 1$ activated agents for x_{new}, it means the region around x_{new} is already known because it is covered by agents. If $L_{\mathcal{A}_i} \in [0, \tau_{\text{good}}]$, it means the prediction of agent $\mathcal{A}_i$ is *good* and it was right to declare itself as an expert on x_{new}. $\mathcal{A}_i$ will therefore seek to generalize in the direction of x_{new} by extending its area of expertise. If $L_{\mathcal{A}_i} \in [\tau_{\text{good}}, \tau_{\text{bad}}]$, then the prediction of $\mathcal{A}_i$ is *inaccurate* but not catastrophic, so it only refines its internal model and keeps its positions waiting for another signal to expand if needed. Then, if $L_{\mathcal{A}_i} \in [\tau_{\text{bad}}, +\infty[$, the prediction of $\mathcal{A}_i$ is *bad,* meaning that it shouldn't have been activated. In reaction to that, $\mathcal{A}_i$ will retract to get away from x_{new}. This rule is the core of agents local self-organization. It allows the agents to move in feature space and refine their model as needed to exclude or include points to better model their close surroundings in response to feedbacks on the quality of their predictions.

Incompetence When there are $N_A = 0$ activated agents and $N \geq 1$ agents that are neighbors of x_{new}, all of the closest agents are candidate on becoming activated on x_{new}. In this situation, if $L_{\mathcal{A}_i} \in [0, \tau_{\text{good}}] \cup [\tau_{\text{good}}, \tau_{\text{bad}}]$, i.e prediction of agent $\mathcal{A}_i$ is *good* or *inaccurate*, then it means that $\mathcal{A}_i$ could become an expert on x_{new} so it refines its model and expands towards x_{new}. Otherwise, if no neighbor gives *good* or *inaccurate* predictions, a new agent centered on x_{new} is created. This rule promotes knowledge reuse and limit redundant agent creation.

Uselessness In practice, we observe a need for agent destruction mechanisms within the system. Indeed, some agents may evolve into degenerate shapes if they receive too much negative feedbacks in a row. They become far too small to be informative. These *dead* agents most probably won't be activated ever again and are no longer useful in the system. Consequently, all agents whose volume falls below a certain threshold value τ_{vol} are destroyed.

Table 3.1 summarize the learning rules governing the behavior of agents in oCELL and Figure 3.2 illustrates the learning process.

	Condition	Agent selected	Action
$N_A = 0$ $N \geq 1$	$L_{\mathcal{A}_i} \in [0, \tau_{\text{good}}] \cup [\tau_{\text{good}}, \tau_{\text{bad}}]$	Neighbor	update model + shape (expand)
	$\forall \mathcal{A}_i,\ L_{\mathcal{A}_i} \notin [0, \tau_{\text{good}}] \cup [\tau_{\text{good}}, \tau_{\text{bad}}]$	Neighbor	create agent
$N_A \geq 1$	$L_{\mathcal{A}_i} \in [0, \tau_{\text{good}}]$	Activated	update shape (expand)
	$L_{\mathcal{A}_i} \in [\tau_{\text{good}}, \tau_{\text{bad}}]$	Activated	update model
	$L_{\mathcal{A}_i} \in [\tau_{\text{bad}}, +\infty[$	Activated	update shape (retract)
$N_A = 0$ $N = 0$	$\varnothing$	$\varnothing$	create agent

Table 3.1: Learning rules of oCELL

Figure 3.2: Flowchart illustrating the learning process

Therefore, oCELL relies on several hyperparameters that must be initialized prior to training. These parameters have an influence on the convergence of the system. These parameters include:

- Initial orthotope size (R): determines the initial confidence region of newly created agents; overly small regions may hinder proper activation and make new agents over-specialize locally.
- Prediction thresholds (τ_{bad} and τ_{good}): define accuracy regimes and influence rule activation frequency.
- Volume variation coefficient (α): controls the rate of expansion or retraction of agents as a fraction of their current volume.
- Destruction threshold (τ_{vol}): specifies the minimum volume below which an agent is considered as a candidate for destruction.

3.1.3 Comparative Study

To evaluate the performances of oCELL, we conducted a comparative study against standard machine learning algorithms on a regression task. This section outlines the experimental protocol and compare performances against various metrics. For this experiment, each context agent employs a linear regression as its internal model. This choice is motivated by the simplicity and transparency of the inner workings of linear transformations. This aligns with the design of oCELL and facilitate clear analysis of the system's behavior.

Experimental Setup

The experiment is conducted on four two-dimensional benchmark functions commonly used in optimization research. These functions were selected due to their nonlinear characteristics and the challenges they present for optimization, particularly for gradient-based methods [26]. This choice is motivated by the fact that state-of-the-art machine learning algorithms [14, 29] partially rely on gradient-based optimization during the learning process.

To ensure relevance of the evaluation while keeping it low-dimensional, we selected classical functions known for their optimization difficulty, characterized by multiple local optima and nonlinear variations [1, 26]. The functions considered are the SSR functions ($f(x, y) = \sin\left(\sqrt{x^2 + y^2}\right)$), Booth, Goldstein-Price, and Beale functions (see Figure 3.3). For each function, a synthetic dataset comprising 8000 observations was generated by uniformly sampling points within the feature space.

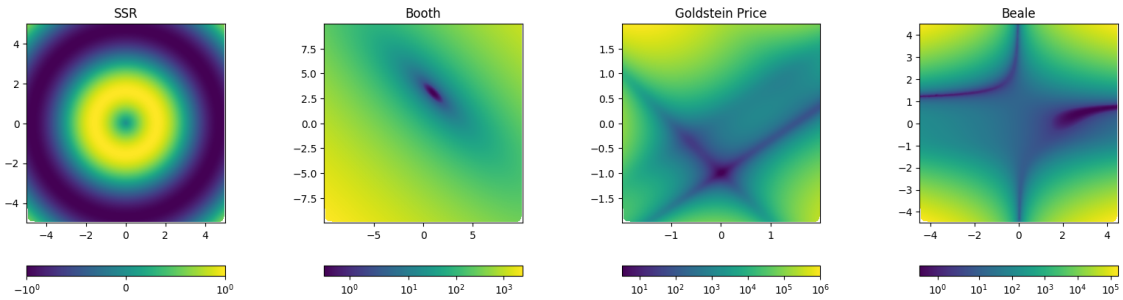


Figure 3.3: Visualization of the 2D functions used to generate the benchmark datasets. The color levels approaching yellow correspond to higher values, while the color levels approaching blue correspond to lower values.

We compare oCELL against widely used ensemble algorithms and common weak learners. The selected weak learners include decision trees [8], SVMs [13], and linear regression [51]. For ensemble methods, we consider both classical approaches such as gradient boosting [22] and random forests [10], alongside state-of-the-art algorithms: XGBoost [14] and LightGBM [29].

All input features are normalized to the range -1 and 1. Hyperparameter optimization is performed via a grid search with the goal of minimizing the mean squared error obtained through 5-fold cross-validation. The resulting best configurations found are reported in table 3.2. These configurations correspond to the parameter sets exposed by the interfaces of the scikit-learn [39], xgboost [14], and lightgbm [29] python libraries. For oCELL, R is a vector corresponding to the initial length on each dimension of the sides of the orthotope for a newly created context agent. The parameters $\tau_{\text{good}}, \tau_{\text{bad}}, \alpha \in \mathbb{R}^3$ correspond to those introduced in section 3.1.2, and *memory_ length* sepcifies the maximum memory size allocated to an agent.

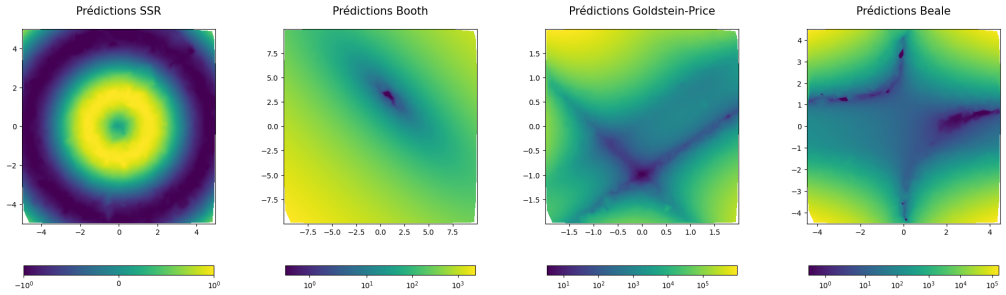


Figure 3.4: Visualization of oCELL predictions on data not included in the training dataset.

Results

The results obtained using the selected evaluation metrics across the considered benchmark functions are reported in table 3.3. oCELL achieves the lowest mean absolute error (MAE) and the highest coefficient of determination (R^2) for all four functions. It also attains the best mean squared error (MSE) on the SSR, Goldstein-Price, and Booth functions, while XGBoost performs slightly better in terms of MSE on the Beale function. Overall, oCELL demonstrates comparable performances to XGBoost and LightGBM for most functions, while consistently outperforming the weak learners. These results indicate that oCELL effectively captures the nonlinear variations of the underlying functions from the data, as illustrated in Figure 3.4.

3.1.4 Explainability and Interpretability

The CELL instantiations are designed to possess native mechanisms for introspection. While its architecture is similar to the localized approximation approach used by techniques such as LIME [42], oCELL achieves intrinsic local explainability through its structural multiagent design.

Interpretability

oCELL's design provides a basis for full model interpretability. When the agents employ inherently transparent models such as linear regression, oCELL itself works as a structurally interpretable white-box model in accordance with the definition introduced in 2.2.3. The system's final prediction is derived from a linear combination of predictions proposed by a restricted, spatially-localized subset of agents. This mechanism simplifies the process of credit assigment compared to traditional global ensemble models (e.g bagging or boosting), which often involve contributions from a large set of weak learners. This property makes oCELL intrinsically interpretable.

Consequently the spatial organization and internal structures of the model can be systematically analyzed to extract essential information about the function underlying the data and its complexity. For didactic purposes, the remainder of this section is based on a training carried out on a synthetic dataset generated from the Beale function (cf. Figure 3.3). This nonlinear function exhibits regimes of amplitude variations that are very different across its domain of definition. The amplitude of the gradients of the Beale function varies greatly at

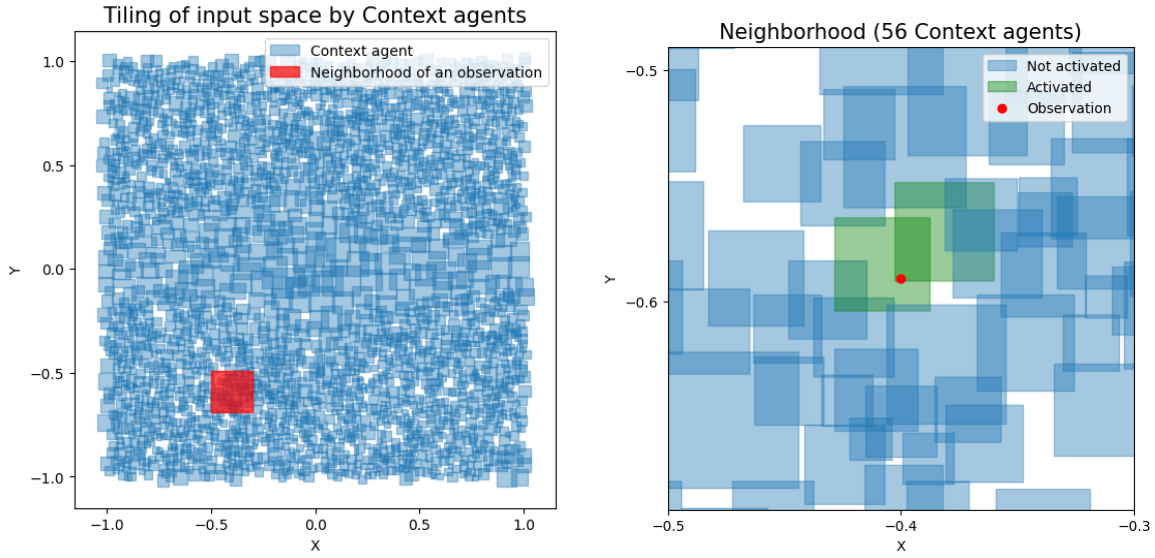
	hyperparameters	SSR	Booth	Goldstein-Price	Beale
oCELL	R	0.1	0.35	0.1	0.05
	τ_{good}	0.01	0.15	0.01	0.1
	τ_{bad}	0.2	0.2	0.2	0.3
	α	0.4	0.1	0.2	0.2
	memory_length	20	50	200	20
XG Boost	learning_rate	0.1	0.05	0.05	0.1
	max_depth	9	9	9	9
	n_estimators	300	300	300	300
	objective	abs. error	abs. error	squ. error	abs. error
Light GBM	learning_rate	0.1	0.1	0.1	0.1
	max_depth	9	5	9	3
	n_estimators	5	5	5	5
	objective	300	300	200	300
	min_child_samples	huber	12	tweedie	tweedie
Random Forest	bootstrap	true	true	true	true
	criterion	abs. error	poisson	poisson	poisson
	max_depth	9	9	9	9
	max_features	null	null	null	null
	min_samples_leaf	1	1	1	1
	min_samples_split	2	2	2	2
	n_estimators	300	200	300	300
Decision Tree	criterion	abs. error	poisson	poisson	poisson
	max_depth	9	9	9	9
	max_features	null	null	null	null
	min_samples_leaf	1	2	2	1
	min_samples_split	2	2	2	2
Gradient boosting	learning_rate	0.1	0.1	0.2	0.2
	loss	abs. error	huber	squ. error	huber
	max_depth	9	9	5	5
	max_features	null	null	null	null
	min_samples_leaf	2	4	1	1
	min_samples_split	2	5	2	2
SVM	epsilon	0.1	0.1	0.1	0.5
	gamma	scale	scale	scale	scale
	kernel	rbf	rbf	poly	rbf
Linear Reg.	fit_intercept	true	true	true	true

Table 3.2: Best hyperparameters set found by grid search on 5-fold cross validation results

		R^2	MSE	MAE
Beale	Decision Tree	0,983	6,94e+06	1,07e+03
	Gradient boosting	0,995	2,08e+06	6,19e+02
	LightGBM	0,996	1,61e+06	4,75e+02
	Linear Reg.	-0,001	4,16e+08	1,19e+04
	oCELL	**0,997**	1,34e+06	**2,53e+02**
	Random Forest	0,995	2,02e+06	5,23e+02
	SVM	-0,139	4,74e+08	8,28e+03
	XGBoost	**0,997**	**1,29e+06**	4,11e+02
Booth	Decision Tree	0,994	1,29e+03	2,60e+01
	Gradient boosting	0,999	1,39e+02	7,44e+00
	LightGBM	0,999	1,38e+02	8,09e+00
	Linear Reg.	0,426	1,17e+05	2,63e+02
	oCELL	**1,000**	**9,52e+00**	**9,91e-01**
	Random Forest	0,998	3,57e+02	1,31e+01
	SVM	0,811	3,89e+04	7,85e+01
	XGBoost	**1,000**	8,72e+01	6,01e+00
Goldstein-Price	Decision Tree	0,994	1,02e+08	5,29e+03
	Gradient boosting	**0,999**	2,37e+07	2,62e+03
	LightGBM	**0,999**	1,80e+07	1,82e+03
	Linear Reg.	0,249	1,22e+10	6,79e+04
	oCELL	**0,999**	**1,21e+07**	**7,02e+02**
	Random Forest	0,998	3,98e+07	3,38e+03
	SVM	-0,127	1,83e+10	5,17e+04
	XGBoost	**0,999**	1,46e+07	1,61e+03
SSR	Decision Tree	0,960	1,89e-02	9,30e-02
	Gradient boosting	0,997	1,22e-03	2,30e-02
	LightGBM	0,998	7,68e-04	1,98e-02
	Linear Reg.	0,000	4,77e-01	6,09e-01
	oCELL	**0,999**	**4,97e-04**	**1,39e-02**
	Random Forest	0,978	1,03e-02	6,48e-02
	SVM	0,972	1,32e-02	7,90e-02
	XGBoost	**0,999**	5,93e-04	1,65e-02

Table 3.3: Comparison between oCELL and the reference algorithms (best scores in bold)

the boundaries of the definition domain ($-4 \leq x, y \leq 4$), while in the central plateau, the variations in gradient amplitude are more subtle. The Beale function is frequently used as a benchmark for testing optimization algorithms [56, 26].



(a) (b)

Figure 3.5: Visualization of the tiling of the space (normalized between -1 and 1) by the context agents for the Beale function (3.5a). Each blue rectangle represents a context agent, and the red rectangle is an example of the neighborhood of an observation. Visualization of the context agents in the neighborhood of an observation (3.5b).

Uncertainty and Confidence

The notion of neighborhood and more broadly the spatial organization of agents allows to extract various metrics of the local organization of agents in the feature space.

Epistemic Uncertainty During training, the agents organize themselves in the feature space. Depending on the training dataset and the complexity of the underlying function, the results of self-organization varies. As illustrated in figure 3.5a, gaps (uncovered zones) in the agent tiling may remain. It is possible that for a given observation, no agent is activated. In such cases, making a prediction requires relying on the proposals of the nearest agents in the observation's neighborhood. To determine if the agents' prediction is reliable, we define the coverage index p_{coverage} of the observation's neighborhood as

$$p_{\text{coverage}} = \frac{V_{\text{agents}}}{V_{\text{neighborhood}}} \tag{3.10}$$

where V_{agents} is the volume covered by the *Context* agents in the neighborhood and $V_{\text{neighborhood}}$ is the volume of the hyperrectangle representing the neighborhood.

Thus, if p_{coverage} is close to 1, the space around the observation contains few gaps: our model is proficient in that region of space. Therefore, it is relevant to consider the predictions of nearby agents. Conversely, if p_{coverage} is close to 0, then the space around the observation contains many gaps. There is a high probability that this region of space was underexplored during training. This may indicate either gaps in the training data specific to that area of space, or that training did not proceed as expected for various reasons (strong non-linearities

in the function to approximate, presence of noise, discontinuity, etc...). We illustrate the utility of the coverage index in a scenario where our system learns from a dataset with an entire portion of missing data (3.6). We observe that the coverage index helps identify the missing portion of data in the training dataset without having directly access to the actual training dataset.

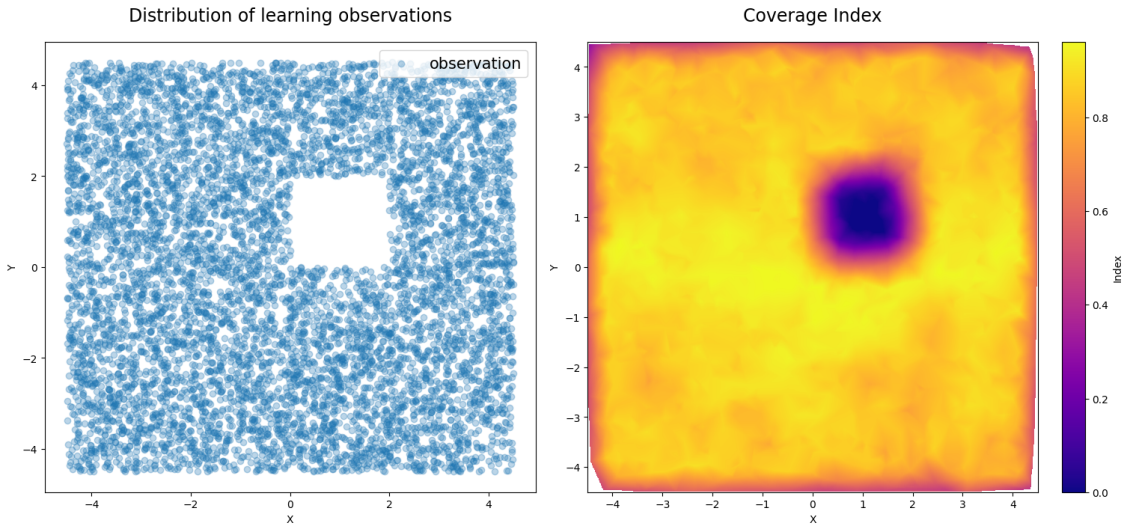


Figure 3.6: Comparison of the distribution map of observations used to train the model (left) and the visualization of the coverage index associated with this model (right).

The covergae index is therefore a quantitative measure of the epistemic uncertainty associated with a prediction. it provides a direct assessment of the model's learned boundaries and limitations in various regions of the feature space. For the development of critical systems, this property is essential to observe in order to assess the risk of extrapolation or interpolation errors.

This measure is fundamentally linked to both explainability and interpretability. Locally, at the individual prediction level, coverage index serves as a crucial component of explanation. By quantifying the reliability of the output, it effectively represents a degree of trustworhtiness and the necessity of relying on neighborhood averaging (cf. 3.1.1). Globally, the aggregated coverage index allows for the systematic analysis and visualization of the entire agent organization across the feature space. This analysis of the overall tiling pattern directly addresses interpretability by giving tools for understanding the model's inherent structure and mechanisms.

Aleatoric Uncertainty After training, when an observation is provided to the model, each activated agent makes a prediction proposal. We can thus calculate the discrepancy between the prediction proposals from the set of proposals of the activated agents ($\mathcal{D}_{\text{proposals}}$) to deduce a measure of aleatoric uncertainty:

$$\sigma_{\text{activated}} = \sqrt{\sum_{p \in \mathcal{D}_{\text{proposals}}} \frac{|p - p_{\text{mean}}|}{N_A}} \tag{3.11}$$

where p_{mean} is the average of the proposals in the set of $\mathcal{D}_{\text{proposals}}$ and N_A is the number of activated agents. This value estimates the disagreement among agents which judge themselves as equally competent to propose a prediction.

This value of discrepancy quantifies the aleatoric uncertainty associated with the model's predictions. It captures the inherent noise and stochasticity as a disagreement among locally activated agents. It's linked to explainability because it primarily relates to locality at the prediction-level, providing useful tools to explain predictions under the angle of trust.

Then, as oCELL is agnostic of the internal model of the agents that compose it, it is possible to use a class of models capable of inherently providing a measure of uncertainty on predictions, such as Gaussian Processes. Those are models which allow estimation of both epistemic and aleatoric uncertainties [45]. Exploiting the strength of this type of model could make the estimation of aleatoric uncertainty more robust. Indeed, several activated agents are needed for $\sigma_{\text{activated}}$ to be informative, using Gaussian Processes would allow to still have a discrepancy value to use even when only one agent is activated. To overcome this drawback, we present later a generalization of this metric so that it can be used to analyze the geometric structures formed by agents during learning.

Variations of the Underlying Function The shape of an agent depends on the region of space it occupies. Moreover, within its confidence area delimited by the orthotope associated to its activation function, the function underlying the data is locally approximable by a simple model. Therefore, in a region containing a large number of agents, variations in the underlying function are likely to be more *difficult* to learn. Thus, we define the density of agents around an observation as

$$\rho = \frac{N_{\text{agents}}}{V_{\text{neighborhood}}} \tag{3.12}$$

where $V_{\text{neighborhood}}$ is the volume of the orthotope representing the neighborhood and N_{agents} is the number of agents intersecting it. Figure 3.7a shows the density of agents in the input variable space for the Beale function. We observe a corridor in the center of the figure for $y \in [-1, 1]$ where the density of agents is low. In the rest of the figure, the density of agents is higher. By examining the distribution of the average volume of agents in the neighborhood of observations (figure 3.7b), we note that areas with high agent density contain smaller agents, and conversely for low-density areas.

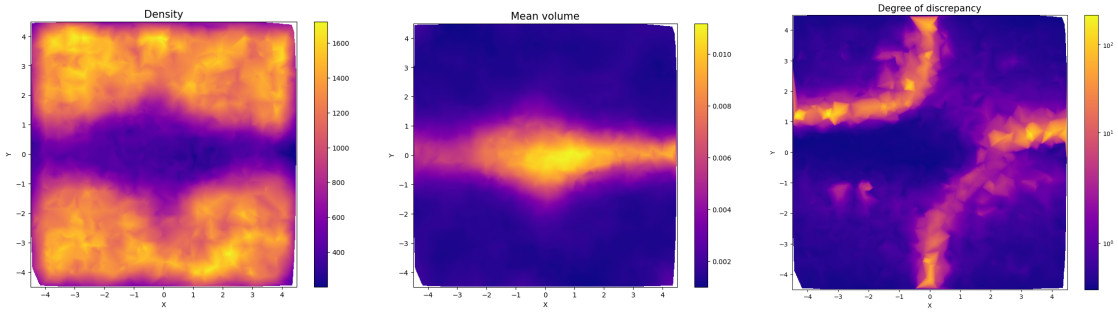
These two regions correspond to very different variation regimes of the Beale function. Indeed, the region containing the largest agents (low density) corresponds to a plateau where the gradient magnitude of the function varies little. In contrast, the region containing the smallest agents (higher density) corresponds to a region of space where the gradient magnitude of the function varies strongly as one approaches the boundary of the definition area (figure 3.7a).

Finally, to complete the analysis, we define the degree of discrepancy κ (which is a derivative of the discrepancy of activated agents $\sigma_{\text{activated}}$ presented earlier) among the predictions of all agents in the neighborhood ($\mathcal{D}_{\text{proposals}}$) as

$$\kappa = \frac{\sqrt{\sum_{p \in \mathcal{D}_{\text{proposals}}} \frac{|p - p_{\text{mean}}|}{N}}}{|y_{\text{mean}}|} \tag{3.13}$$

where N is the number of agents in the neighborhood, and y_{mean} is the average of predictions from the agents in the neighborhood.

If κ is close to 0, then the predictions of the agents in the neighborhood are very close to each other. This indicates that the underlying function of the data varies little around the observation. Conversely, if its value is high, then the agents in the neighborhood make very different predictions, suggesting that the regime of variation of the underlying function is likely to change abruptly in that area. The degree of discrepancy helps identify regions in the input space where approximating the underlying function is most challenging. For the Beale function, the degree of discrepancy highlights regions of the space with the most complex variations (see figure 3.7c where the yellow areas represent the highest disagreement between agents).



(a) (b) (c)

Figure 3.7: Visualization of density (3.7a), average volume (3.7b), degree of discrepancy (3.7c) of *Context* agents.

The metrics presented above allow to extract measures of epistemic and aleatoric uncertainty on predictions, as well as information on function variations based on the shape and position of agents in the feature space.

We have extracted measures of epistemic (coverage index) and aleatoric (activated discrepancy) uncertainty from the predictions of the system as well as information about the variations of underlying function from the shape (volume and density) and spatial organization of agents in the feature space (degree of discrepancy). These introspection tools reveal capabilities that are useful for various purposes such as fault detection, model debugging, or smart resampling strategies to improve learning in specific, uncertain regions of the feature space. Finally, Table 3.4 summarizes the scopes and links between the defined metrics and the notions of explainability and interpretability.

Metric	Scope	Explainability	Interpretability
Coverage Index (p_{coverage})	Local + Global	✓	✓
Activated Discrepancies ($\sigma_{\text{activated}}$)	Local	✓	×
Density (ρ)	Local + Global	✓	✓
Degree of Discrepancy (κ)	Local + Global	✓	✓

Table 3.4: Summary of the metrics introduced and their link to explainability or interpretability

3.1.5 Limitations

We demonstrated that oCELL can address supervised learning tasks, but several limitations remain, which are tackled partly in section 3.2.

A first limitation concerns the neighborhood prediction mechanism. Our current design assumes that an agent possesses uniform knwoledge throughout its activation region (the orthotope associated with its activation function). In practice, an agent is typically more reliable near its centroid and less accurate near the boundaries of its region. Despite this, all agents contributing to a prediction currently receive equal weight. This is an artifact of the uniform-knowledge assumption. A more principle approach would weight each agent according to its estimated local expertise level at the query point.

We explored this idea by weighting predictions using the Euclidean distance between the input and each agent's centroid, but this did not improve performance, likely because centroid distance alone does not reflect the whole geometry of each orthotope. A more appropriate distance measure should incorporate both the agent's centroid and the shape of its support. As illustrated in 3.8 (side by side comparison of an overgeneralizing agent and a representative agent by visualizing their respective orthotopes (represented by a blue rectangle) and training data (represented by red dots)), an agent may occupy a large orthotope while its training data populate only a small manifold (i.e a subregion), causing spurious known-region responses and false positives. Accurate spatialization and potentially richer base shapes is therefore essential to capture local data structures and prevent overgeneralization.

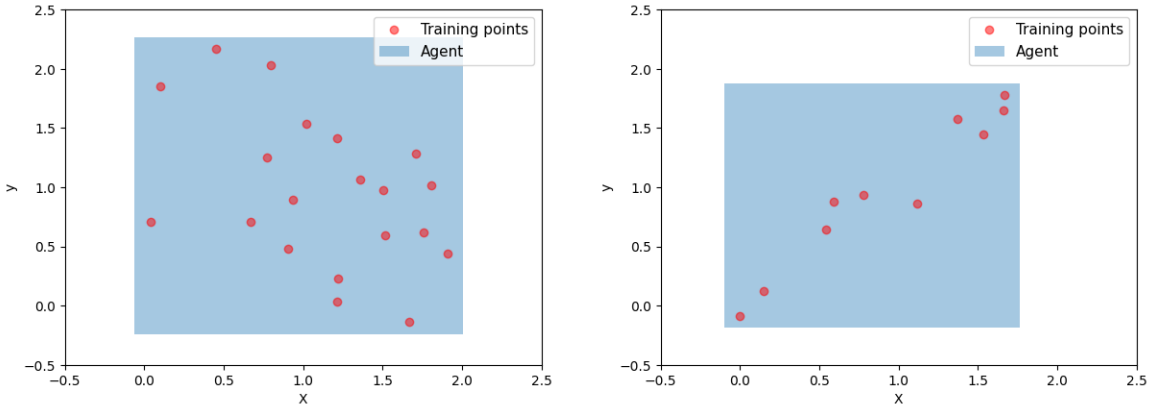
Furthermore, we observed that oCELL's agent destruction mechanism was inadequate. Agent overproduction was a frequently observed phenomenon, occasionally resulting in significant local redundancy and the persistence of harmful parasitic agents within the collective. This necessitates the development of more sophisticated destruction strategies explicitly designed to enforce local cooperation. In addition to that, the current single-point agent creation scheme seems too limited, as it directly contributes to local redundancy (slow local convergence). It should be more stable to initialize agents using multiples points to minimize local overlap and make local convergence faster.

Moreover, we observed that oCELL's agent destruction system was not sufficient and that agent overproduction was fairly common, sometimes leading to high local redundancy and the persistence of agents that parasitize the collective without improving it. This highlights a need for smarter destruction mechanisms that are designed towards local cooperation and

more efficient agent creation schemes because creating agents with only one point can cause high local redundancy.

Furthermore, although oCELL is designed for online learning, it treats incoming samples as independent, failing to exploit temporal correlations in data streams. This limits its sample efficiency and reduces its ability to adapt to evolving distributions.

Finally, some hyperparameters remain difficult to tune. In particular, the initial side lengths of newly created agents have a disproportionate impact on performance and require specifying a scale per feature dimension. Similarly, the error thresholds used to classify predictions as *good*, *inaccurate* or *bad* are hard to tune properly to enable an efficient balence in the triggering of the various adaptation mechanisms or agents (i.e learning rules).



(a) Representative orthotope

(b) Overgeneralizing orthotope

Figure 3.8: Side by side comparison of representative 3.8a agent against a overgeneralizing agent 3.8b. The blue rectangle corresponds to the orthotope associated to the activation function of the Context agent. Each red dot corresponds to a training point seen by the Context agent.

Taken together, these limitations expose a broader structural issue. oCELL relies on spatialization and prediction aggregation mechanisms that are too rigid. The orthotope representation is too coarse to support reliable interpolation when agents overlap or specialize unevenly. These limitations also highlight the need for inference machanisms that weight agents according to their estimated expertise level rather than using discrete activations. These considerations motivate the design of a new CELL system in which spatialization, inference and adaptation are grouned in smoother data representations.

3.2 kCELL: Multiagent Ensemble Learning with Kernel Spatialization

Building upon the limitations of oCELL, we introduce kCELL (kernel CELL), the second instantiation of the CELL paradigm. With oCELL, we demonstrated that a population of agents can achieve competitive performance and provide valuable introspection signals such

as epistemic and aleatoric uncertainty estimates and qualitative information about local function variations. However, its rigid spatial representation ultimately limits its ability to build reliable and adaptive regions of expertise. This constraint becomes problematic when there is need for fast adaptation in an online settings.

kCELL addresses these issues by replacing orthotope-based spatialization with a kernel-based representation. Each agent has its activation function in the form of a RBF kernel that defines a smooth continuous distance between input features and agents, drawing inspiration from RBF Networks [33] and MoE (Mixture of Experts) [55] gating mechanisms. This change allows the system to express spatial structures more finely, with more degrees of freedom than what is permitted with orthotopes, thus breaking the assumption of knowledge uniformity over the region of expertise of oCELL.

In addition to that, kCELL introduces a smooth aggregation rule in which agents contribute to predictions proportionally to their kernel-defined relevance, i.e proportionally to their estimated expertise level. This brings the inference closer to the probabilistic weighting strategies used in Gaussian Processes [46].

For those reasons kCELL improves interpolation quality, agents' data representations and yield more stable learning dynamics that are less dependent on brittle hyperparameters, which is better suited for online adaptive learning. kCELL is a more expressive and robust extension of the CELL paradigm that also preserves the explainability properties demonstrated with oCELL.

First, Section 3.2.1

First, Section 3.2.1 describes the internal structure of context agents in kCELL, including their spatialization in the feature space and their prediction mechanisms. Then, Section 3.2.2 introduces the learning rules that cover agent creation, adaptation and self-organization leading to the emergence of learning in the system. Section 3.2.3 presents a study of kCELL's capabilities in terms of adaption in a non-stationary environment leveraging the introspection tools specific to the models derived from CELL. Subsequently, section 3.2.6 discusses the key differences between oCELL and kCELL and details the limitations of kCELL.

3.2.1 Context Agents

As for oCELL , the main entities of kCELL are the Context agents. A Context agent $\mathcal{A}_i$ is characterized by an activation function $\phi_i(x)$ and a prediction function $f_i(x)$ such as

$$\mathcal{A}_i = \{\phi_i, f_i\}$$

However, unlike in oCELL, the Context agents of kCELL are spatialized differently. In oCELL, a binary activation (the point is inside or outside the associated orthotope) is used while in kCELL, a RBF kernel is used as a continuous and smooth activation function to represent the area of expertise in feature space.

First, the spatialization mechanism of context agents based on RBF kernels is described. Then the soft weighted prediction mechanism allowing for weighting based on estimated level of expertise is presented.

Kernel Spatialization

In kCELL, contrary to using orthotopes a RBF kernel is used as the activation function to unlock more degrees of freedom to represent the areas of expertise . This activation function is smooth and differentiable, making the system more optimization-friendly to be used for control tasks as a dynamics model. Therefore, an agent is spatialized by the mean $\mu_i \in \mathbb{R}^n$ and covariance matrix $\Sigma_i \in \mathbb{R}^{n \times n}$ of the distribution of its training points. Since this activation function has statistical significance, it is easier to define the neighborhood as a confidence interval whose confidence value can be adjusted.

Figure 3.9: Comparison between area of expertise oCELL vs kCELL

Definition 4. The distance between a point $x \in \mathbb{R}^n$ and a Context agent $\mathcal{A}_i$ is defined as the mahalanobis distance:

$$d(\mathcal{A}_i, x) = D_M(\mu_i, \Sigma_i, x) = \sqrt{(x - \mu_i)^\top \Sigma_i^{-1} (x - \mu_i)}$$

Definition 5. A Context agent $\mathcal{A}_i$ is considered as a neighbor of $x \in \mathbb{R}^n$ if x is likely to belong to the training dataset of $\mathcal{A}_i$ such as

$$D_M(\mu_i, \Sigma_i, x)^2 \leq \chi^2_{n,0.95}$$

where $\chi^2_{n,0.95}$ denotes the 95th percentile of the chi-squared distribution with n degrees of freedom.

Definition 6. The activation function $\phi_i : \mathbb{R}^n \mapsto [0, 1]$ of a Context agent $\mathcal{A}_i$ for a point $x \in \mathbb{R}^n$ is given by the RBF kernel value

$$\phi_i(\mathcal{A}_i, x) = \exp\left(-\frac{d(\mathcal{A}_i, x)}{2l^2}\right)$$

with l the lengthscale of the kernel.

The activation score of a point x for a given agent $\mathcal{A}_i$ can be interpreted as the likelihood that the agent has previously been trained on points similar to x, and thus reflects the agent's degree of knowledge over the corresponding region of feature space. We can draw a parallel between this activation function based on RBF kernel and the coverage index , an explainability metric derived from oCELL which is also linked to the degree of knowledge of a point. In kCELL, this explainability property related to the knowledge of a point emerges naturally from the definition of the system.

One of the requirements of kCELL is to be lightweight, meaning that as few points as possible should be retained in memory to truly represent information through local models. As previously stated, the spatialization of each agent can be fully described by a mean μ_i and a covariance matrix Σ_i which are the parameters of the activation function ϕ_i. Each time a point is ingested by agent $\mathcal{A}_i$, its mean and covariance matrix are updated using the Welford's algorithm [52] such as

$$\begin{aligned} \mu_{i\,|\,t+1} &= \mu_{i\,|\,t} + \frac{x - \mu_{i\,|\,t}}{n+1} \\ \Sigma_{i\,|\,t+1} &= \frac{1}{n}\left(\Sigma_{i\,|\,t+1} + \left(x - \mu_{i\,|\,t}\right)\left(x - \mu_{i\,|\,t+1}\right)^\top\right) \end{aligned}$$

where n is the number of points ingested by the agent to update its shape. With this approach the shape change is expressed purely as a mean and covariance estimation problem.

Soft-Weighted Prediction

In kCELL, we get rid of the knowledge uniformity hypothesis of agents to better represent the knowledge of the system. Therefore the activation function of a context agent materializes its area of expertise through a smooth RBF kernel. The further a point is from the agent's centroid, the lower its activation value. This is because we assume that expertise is maximized at the agent's center. Agents with higher activation values contribute more heavily to the final output because they are more expert than others at predicting.

Let $S_k = \{i | \mathcal{A}_i \text{ is in the } k\text{-closest to } x\}$ denote the index set of the k nearest agents in feature space. The final prediction $f(x) \in \mathbb{R}^m$ of the system is then given by

$$f(x) = \sum_{i \in S_k} w_i(x) f_i(x)$$

where the normalized contribution weights $w_i(x)$ are defined as

$$w_i(x) = \frac{\phi_i(x)}{\sum_{j \in S_k} \phi_j(x)}$$

This formulation ensures that each contributing agent's influence on the final prediction is proportional to its estimated competence.

3.2.2 Learning Rules

As in oCELL, learning rules are designed around a set of actions and conditions that triggers those actions to digest x_{new}, y_{new} that are fed to the system sequentially as a data stream. In context agent learning systems, the set of actions consists of the actions that can be performed individually by agents such as updating their model or shape; and more meta-level actions such as destroying or creating new agents. These actions are triggered by conditions that generally depend on two factors: the number of current neighbors and a feedback value calculated from the proposals of selected agents.

When there are $N > 1$ agents that are neighbors of x_{new}, all of those agents are theoretically considered as experts to predict for x_{new}. As the final prediction fo the system $f(x) = \hat{y}$ is computed from the proposals of all those agents, the agents need to cooperate together locally to improve the local knowledge. For this purpose, we define $\Delta_{\mathcal{A}_i}$, the fractional change in error when leaving agent $\mathcal{A}_i$ out of the prediction process,

$$\Delta_{\mathcal{A}_i} = \frac{E_{-i} - E}{E} = \frac{|\hat{y}_{-i} - y_{new}| - |\hat{y} - y_{new}|}{|\hat{y} - y_{new}|}$$

where E is the prediction error, $E_{-i} = |\hat{y}_{-i} - y_{new}|$ is the prediction error without considering the proposition of prediction of agent $\mathcal{A}_i$.

So, if $\Delta_{\mathcal{A}_i} > 0$ then it means that removing agent $\mathcal{A}_i$ from the prediction group had a negative impact on the prediction error, meaning that agent $\mathcal{A}_i$ has a positive contribution

to reducing the error locally and conversely for $\Delta_{\mathcal{A}_i} < 0$. This value indicates for a given point, which agents are strong and which are weak in predicting for x_{new}. We can therefore consider that weak agents need to improve their local model, while strong agents are already good and need to strengthen their local anchoring at the given point. This update rule allows for the continuous improvement of the group locally by improving the weakest agents while keeping them mobile, thus allowing them to position themselves elsewhere if needed.

When there is only one agent ($N = 1$) that is neighbor to x_{new}, it can't be updatd according to the same rules. Indeed, its contribution to the error cannot be compared to the ones of other neighors (as if $N > 1$). Therefore, we define $\Delta'_{\mathcal{A}_i}$, the relative error reduction compared to a baseline prediction given by a running short term linear predictor

$$\Delta'_{\mathcal{A}_i} = \frac{E_{base} - E_i}{E_{base}} = \frac{|y_{base} - y_{new}| - |\hat{y}_i - y_{new}|}{|y_{base} - y_{new}|}$$

where E_{base} is the prediction error of the short term linear predictor, $E_i = |\hat{y}_i - y_{new}|$ is the prediction error of agent $\mathcal{A}_i$.

So, if $\Delta'_{\mathcal{A}_i} > 0$ then it means that $\mathcal{A}_i$ outperforms the short term linear predictor locally around x_{new}, giving indications that agent $\mathcal{A}_i$ might be useful locally, and conversely for $\Delta'_{\mathcal{A}_i} < 0$. This value allows to estimate how expert the agent is relative to the short-term baseline. We can therefore consider that if the only neighbor agent is beaten by the baseline, then it probably shouldn't be a neighbor to this point. It should skip interacting this x_{new} and y_{new}, waiting to reposition itself by ingesting new points, to be destroyed, or for another agent to become a neighbor in that area so they can improve together. Conversely, if it is better than the baseline, then it is a local expert whom should stay in that area since it represents the only local source of knowledge. So, it should improve its local model and strengthen its local anchoring around x_{new}. This update rule allows single neighbors to specialize locally even when alone in an area.

Finally if no agent is considered a neighbor ($N = 0$), $\Delta'_{\mathcal{A}_i}$ is computed since as in the case $N = 1$, there is no neighbor to compare to. Thus, if $\Delta'_{\mathcal{A}_i} > 0$, it means that the neared agent is more relevant than the short term linear predictor. Therefore, it seems that this agent should encompass the point and become its neighbor by improving its model and extending its shape towards x_{new}. On the other hand, if $\Delta'_{\mathcal{A}_i} < 0$, it means that even the nearest agent is not relevant for prediction and therefore the system probably has no knowledge of x_{new} and should add it to its knowledge base by covering the space around it by creating a new agent. To avoid initializing the new agent with only one point (as in oCELL) and accelerate local convergence, it is initialized from a short term buffer containing last points seen by the system.

Since agents are created, a destruction mechanism is needed. An, agent might position poorly in the feature space or choose to ingest the wrong points making its model no longer representative of the covered area. Decision errors happen almost all the time when working in an online setting because it is impossible to know what the future points will look like. It can only be guessed from local relationships between successive points. If no destruction mechanism is introduced to mitigate the proliferation of agents, the number of agents will grow indefinitely and performances will be hurt by bad agents, preventing local specialization in the system and ultimately dragging down the predictive accuracy.

Therefore, we define instantaneous normalized confidence $c_{\mathcal{A}_i} \in [-1, 1]$ of an agent as the various feedbacks received by the agent

$$c_i = \tanh\left(k \times \begin{cases} \Delta_{\mathcal{A}_i} & \text{if } N > 1 \\ \Delta'_{\mathcal{A}_i} & \text{else} \end{cases}\right)$$

where k is the steepness of the tanh function.

Then, we define $\bar{C}_{\mathcal{A}_i} \in [-1, 1]$, the running confidence of agent $\mathcal{A}_i$ as the exponental moving average of successive instantaneous confidence values (i.e feedback values) received by the agent

$$\bar{C}_{\mathcal{A}_i \,|\, t+1} = (1 - \lambda)\, \bar{C}_{\mathcal{A}_i \,|\, t} + \lambda c_{\mathcal{A}_i}$$

where λ the smoothing factor.

The confidence value $\bar{C}_{\mathcal{A}_i}$ is calculated for each agent $\mathcal{A}_i$ and updated each time $\mathcal{A}_i$is selected as a neighbor to x_{new}. It allows to track its performance over the course of successive updates. When $\bar{C}_{\mathcal{A}_i} > 0$, it means that, on average, $\mathcal{A}_i$ is categorized as strong compared to other neighbors or the short-term baseline predictor, and conversely if $\bar{C}_{\mathcal{A}_i} < 0$. Confidence this allows us to distinguish between agents that strengthen the system and those that degrade it. We deduce that an agent whose trust falls below a certain threshold $\tau_{\text{confidence}}$ should be destroyed to allow the emergence of new and more efficient local structures.

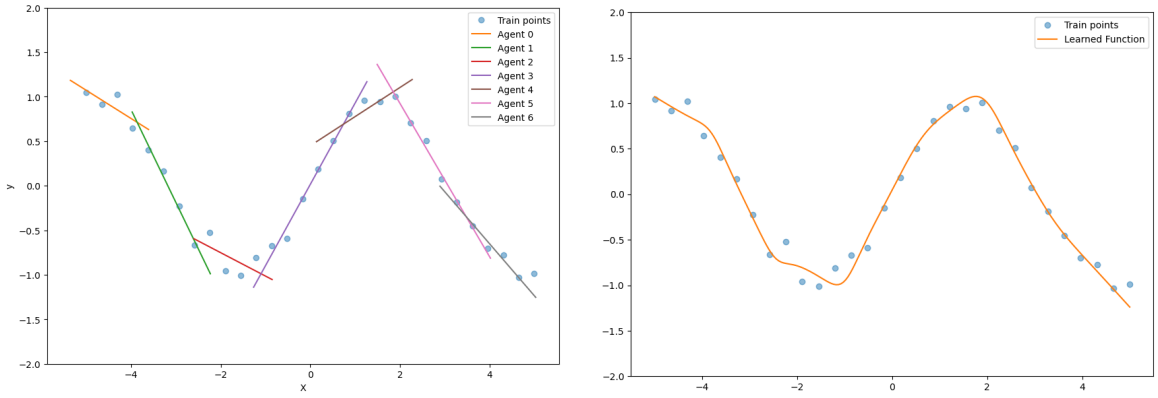
Finally, Table 3.5 presents a summary of the learning rules that describe the behavior of context agents in kCELL.

	Condition	**Agent selected**	**Action**
$N = 0$	$\Delta'_{\mathcal{A}_i} > 0$	Closest	update model + shape
	$\Delta'_{\mathcal{A}_i} < 0$	Closest	create agent
$N = 1$	$\Delta'_{\mathcal{A}_i} > 0$	Neighbor	update model + shape
	$\Delta'_{\mathcal{A}_i} < 0$	Neighbor	$\varnothing$
	$\bar{C}_{\mathcal{A}_i} \leq \tau_{\text{confidence}}$	Neighbor	destroy agent
$N > 1$	$\Delta_{\mathcal{A}_i} > 0$	Neighbor	update shape
	$\Delta_{\mathcal{A}_i} < 0$	Neighbor	update model
	$\bar{C}_{\mathcal{A}_i} \leq \tau_{\text{confidence}}$	Neighbor	destroy agent

Table 3.5: Learning rules of kCELL where N is the number of neighbors, $\Delta'_{\mathcal{A}_i}$ the relative error reduction compared to a baseline prediction and $\Delta_{\mathcal{A}_i}$the fractional change in error when leaving agent $\mathcal{A}_i$ out of the prediction process.

To visualize the result of a learning and local cooperation of agents, we use a 1-dimensional example. We generate a small synthetic dataset from the function $f(x) = \sin(x) + \epsilon\mathcal{N}(0, 1)$ with $\epsilon \in \mathbb{R}$ the noise scaling factor. We simulate online learning by sequentially feeding the agent one x_{new}, y_{new} tuple at a time. Each point is only seen once during learning as this is an important property of an effective and rapid online learning algorithm especially useful in real time dynamics modeling. Figure 3.10 presents a side-by-side visualization of the spatial organization of agents (3.10a) against the result of aggregating their predictions (3.10b). We can observe that the agents overlap and do not fit perfectly across their entire area of expertise; at the edges of agents the fit seems to have more errors and conversely on

the center. This is the expected behavior resulting from the assumption of non-uniformity in the knowledge within areas of expertise. Finally, we observe that the overall predictions approximate closely the function considering the noise added to the data. It demonstrates kCELL's ability to generalize effectively by interpolation. This highlights the usefulness of a weighted aggregation function as presented in 3.2.1.



(a) Spatial organization

(b) Resulting prediction

Figure 3.10: Visualization of local cooperation of agents in predicting a noisy sinus function. (3.10a) presents the spatial organization of agents where is colored segment represents the mapping of an individual agent between feature space and output space. (3.10b) shows the resulting predictions obtained from the aggregation of agents local models, illustrating the interpolation capabilities of the model.

3.2.3 Adaptation to Non-Stationary Dynamics

In kCELL, the agents learn a function from a stream of data through local modeling and cooperation. In this section we present an experiment and introspection studies to demonstrate the capabilities of kCELL in terms of online adaptation and raw performances in online non-stationary dynamics modeling.

Experimental Protocol

To evaluate the capabilities of kCELL to adapt online to non-stationary dynamics, we conducted experiments on two MuJoCo [50] simulation environment: InvertedPendulum and Hopper. For each environment, datasets were collected under three gravitational acceleration values (*default* gravity $g = 9.81$, *low* gravity $g = 4$, *high* gravity $g = 20$) using pretrained reinforcement learning agents trained with Soft-Actor-Critic (SAC) algorithm [24] (with stable-baselines3 implementation [41]). Training data were streamed in the same order (*default* $\rightarrow$ *low* $\rightarrow$ *high*) to simulate abrupt changes in the underlying system dynamics. The characteristics of the generated datasets are presented in Table 3.6.

The environments were selected to contrast a low-dimensional setting (InvertedPendulum with 4-dimensional states and 1 action) with a higher-dimensional one (Hopper with 11-dimensional state and 3 actions). Mahalanobis distance values become larger the higher the

number of dimensions. Thus we want to study the impact of increasing dimensionality on the representativity of the spatialization of kCELL which was one of the main limitations of oCELL (cf 3.1.5). We also compare kCELL to Adaptive Hoeffding Trees [4] which is an efficient adaptive online learning method based on decision trees designed to handle concept drift.

Environment	State Size	Nb Actions	Dynamic Changes	Nb Learning Steps
InvertedPendulum	4	1	$g \in \{9.81, 4, 20\}$	30k
Hopper	11	3	$g \in \{9.81, 4, 20\}$	50k

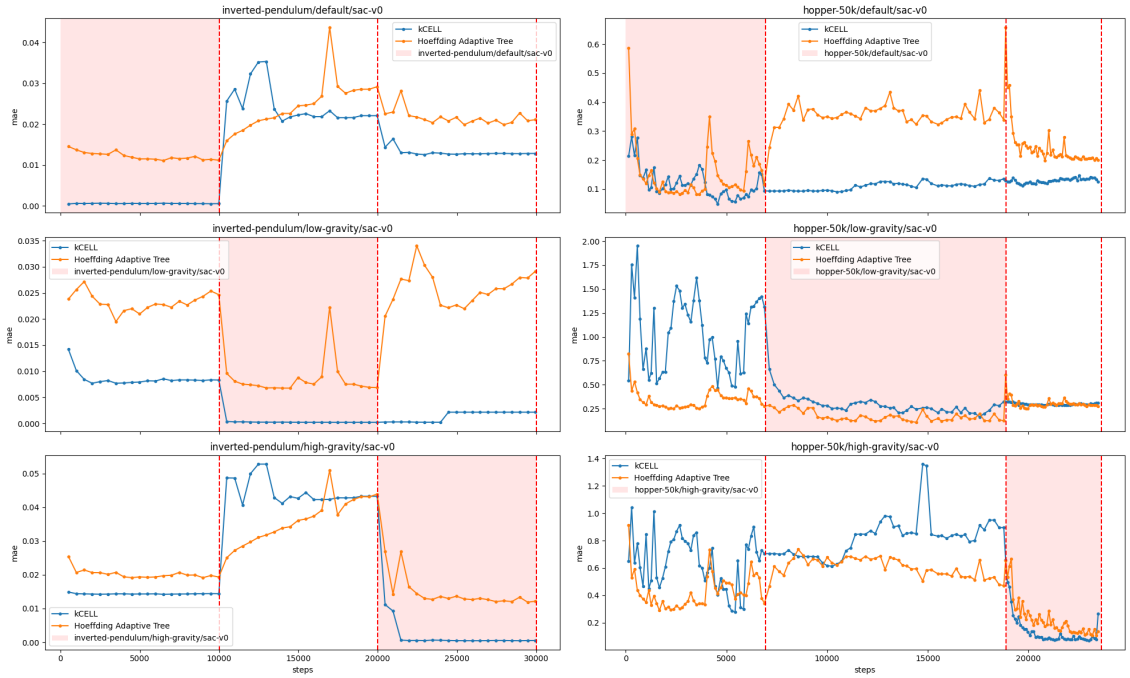
Table 3.6: Characteristics of datasets generated with different values of g. *State Size* stands for the number of features in states, *Nb Actions* stands for the number of continuous actions, *Dynamic Changes* corresponds the set of g values used to generate the datasets, *Nb Learning Steps* corresponds to the budget in number of training steps used to train the RL agent used to generate each dataset (one RL Agent per dataset).

Prediction Error Analysis

Figure 3.11 illustrates the evolution of Mean Absolute Error (MAE) for kCELL and Adaptive Hoeffding Trees algorithms measured on the test sets associated with each value of g for each environment. Across both environments, the MAE consistently decreases on the test set corresponding to the currently observed gravity value, indicating effective online adaptation to changes in dynamics (refer to semi-transparent red areas in Figure 3.11).

For the InvertedPendulum environment (cf. Figure 3.11a), kCELL systematically outperforms Adaptive Hoeffding Trees when the training and testing gravity values coincide, exhibiting lower MAE throughout each stationary phase. This suggests that the local spatialized modeling strategy employed by kCELL allows more accurate approximation of the underlying dynamics in low-dimensional settings. For the Hopper environment (cf. Figure 3.11b), both approaches achieve comparable MAE across the different gravity regimes. However, we notice that the error curve of kCELL displays higher variance, reflecting the self organization mechanisms of agents (creation, destruction, updates). This could also be due to numerical out-of-distribution prediction instability of CELL approaches which is a known problem. Indeed, due to spatialization, kCELL is not designed to predict too far outside their domain of expertise leading to very low activation scores for far away points and potential numerical instabilities.

For kCELL, we notice that for InvertedPendulum, transitions between gravity values lead to an increase in MAE on past test sets, seemingly reflecting partial forgetting of earlier dynamics. In contrast, for Hopper, kCELL exhibits stable MAE on previously encountered gravity values even after multiple regime changes, suggesting retiention of prior knowledge.



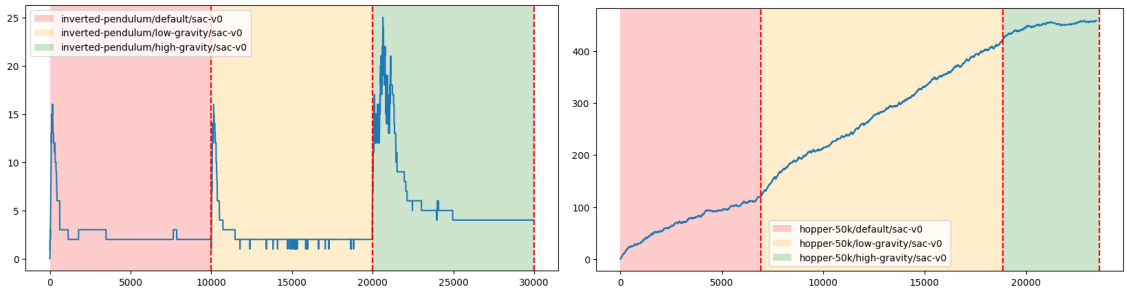
(a) InvertedPendulum (b) Hopper

Figure 3.11: Evolution of test error on each test dataset. Each horizontal plot shows the evolution of error on a test dataset (in order top to bottom: $g = 9.81$ (*default*), $g = 4$ (*low gravity*), $g = 20$ (*high gravity*). The semi-transparent red area corresponds to the training period in which the training and test sets corresponds to the same dynamic variations. The dotted vertical red lines corresponds to changes in dynamics i.e to a change of train dataset with a different value of g. The x axis corresponds to the number of point ingested (number of steps) and the y axis corresponds to the MAE over the corresponding training set. The blue curve corresponds to kCELL's MAE measurements. For example the top plot corresponds to the MAE calculated on test data for $g = 9.81$ and red area corresponds to the training phase in which $g = 9.81$

Structural Evolution of Agent Population

We analyze the evolution of agent population during training illustrated in Figure 3.12. As previously hinted by MAE analysis, we notice contrasting dynamics in the population evolution during learning between the two environments. In InvertedPendulum (cf. Figure 3.12a), each gravity change induces a short-term surge in the number of agents, followed by a reduction and stabilization around a nominal value. This indicates a restructuring of the population, where novelty triggers agent creation and confidence-based mechanisms triggers destruction to eliminate less relevant agents.

In Hopper (cf. Figure 3.12b), the number of agents grows monotonically throughout training. Distinct growth regimes are observed for each gravity value with first a logaritmic growth ($g = 9.81$), second a seemingly linear growth ($g = 4$) and third a slight logarithmic growth ($g = 20$) with stabilization around 20k steps. This accumulation of agents suggests a continuous detection of local dynamics without systematic removal of existing agents.



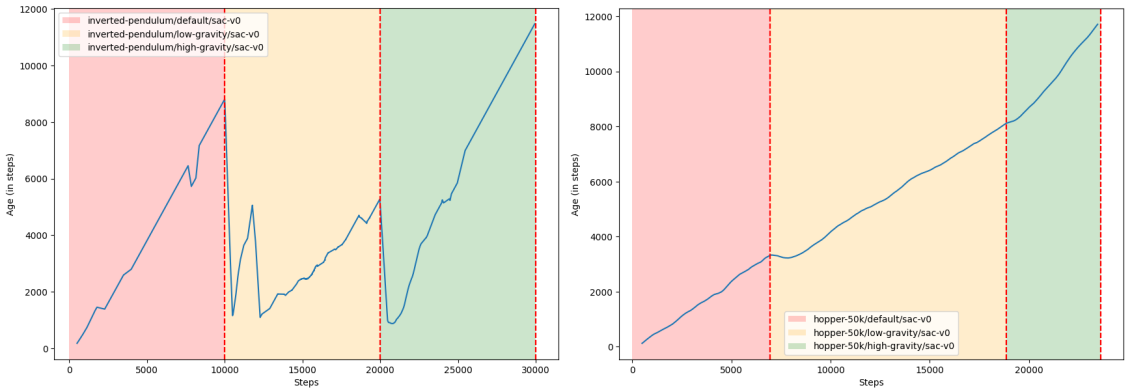
(a) InvertedPendulum (b) Hopper

Figure 3.12: Evolution of the number of agents in kCELL during learning. Each semi-transparent colored area corresponds to a given training dataset / dynamic variation regime. The dotted vertical red lines corresponds to changes in dynamic i.e to a change of train dataset with a different value of g. The x axis corresponds to the number of point ingested (number of steps) and the y axis corresponds to the number of agents in the system.

Agent Age and Confidence Dynamics

The mean age of agents provides further insights on kCELL as presented in Figure 3.13. In InvertedPendulum (cf. Figure 3.13a), gravity changes coincide with abrupt drops in mean agent age. This means that the population becomes suddenly younger, which is consistent with the replacement of older agents by newly created ones following a shift in dynamics. This behavior aligns with a confidence-driven selection mechanism that favors agents specialized to the current dynamics.

In Hopper (cf. Figure 3.13b) however, the mean agent age increases almost constantly, with only minor decreases follwoing the first gravity change and a slight slowdown after the second. This indicates that older agents persist over time and are not fully displaced by newer ones, despite agent creations.



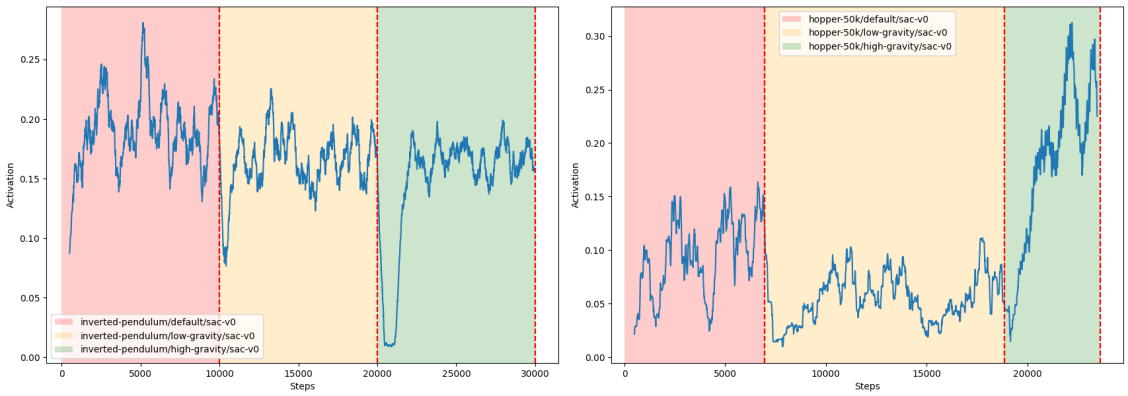
(a) InvertedPendulum (b) Hopper

Figure 3.13: Evolution of the age of agents in kCELL during learning (smoothed with a window of 500 steps). Each semi-transparent colored area corresponds to a given training dataset / dynamic variation regime. The dotted vertical red lines corresponds to changes in dynamic i.e to a change of train dataset with a different value of g. The x axis corresponds to the number of point ingested (number of steps) and the y axis corresponds to the mean age of agents (in steps) in the system .

Agent Activation and Local Expertise

Figure 3.14, shows the evolution of the activation of closest agent during learning. In InvertedPendulum (cf. Figure 3.14a), each gravity change results in a sharp drop in activation, signaling a loss of relevance of existing local modals. Activation values subsequently recovers as new agents acquire expertise under the new dynamics.

In Hopper (cf. Figure 3.14b), activation levels differ across the different gravity regimes, with distinct nominal mean activation value observed for each values of gravity acceleration. Slight activation decreases are detectable immediately following changes in dynamics, these effects are less visible than in InvertedPendulum and partially blurred by higher variance regimes.



(a) InvertedPendulum (b) Hopper

Figure 3.14: Evolution of the maximum activation value of agents in kCELL during learning (smoothed with a window of 500 steps). Each semi-transparent colored area corresponds to a given training dataset / dynamic variation regime. The dotted vertical red lines corresponds to changes in dynamic i.e to a change of train dataset with a different value of g. The x axis corresponds to the number of point ingested (number of steps) and the y axis corresponds to the maximum activation value of agents.

In the higher-dimensional Hopper environment, agent persistence can be explained by two non-exclusive mechanisms. First, it is possible that agents trained under earlier gravity settings retain partial competence in overlapping regions of the state-action space. Those agents are then incrementally adapted as new data arrives leading to multi-regime reuse. In this case the growth of agents can be explained mainly by the modeling complexity introduced by the change in dynamics. Second, high-dimensional effects may reduce the sensitivity of the Mahalanobis distance, yielding sparse activation of older agents that prevents both their meaningful adaption and confidence degradation. In this case, those agents can be considered as frozen because they are too tightly tied to previously visited regions that do not overlap with newly discovered ones after dynamic changes.

To disambiguate whether agent persistence in the Hopper environment arises from adaptive reuse or from limited cross-regime activation, we analyze the activation patterns of the final agent population obtained after full training. Figure 3.15 presents a heatmap of agent activations for all training samples, where columns correspond to agents sorted by age (from oldest to youngest) and rows correspond to training data points (in order) aggregated into bins. The resulting heatmap exhibits distinct block patterns, with older agents primarily activated for samples from the earliest dataset and progressively younger agents ddominating activation for later datasets. We can also observe that the shape of the number of agents curve (presented in Figure 3.12b) appears in the heatmap drawn by points where agents are the most active. This organization further indicates that agents remain selectively active for the dynamics under which they were trained and created rather than becoming inactive or obsolete.

Complementary, we provide a two-dimensional UMAP [36] projection of the training data in Figure 3.16, with samples colored according to their corresponding gravity regime. The projection reveals a clear separation between training datasets, suggesting that changes

in dynamics induce distinct regions in the state-action space. This structural separation provides a plausible geometric explanation for the coexistence of multiple generations of agents. Because data from different dynamics regimes occupy largely disjoint regions, agents specialized to earlier dynamics remain relevant for their corresponding subspaces without interfering with those created for later regimes.

These analyses support the claim that agent retention in the Hopper environment is primarily driven by regime-specific specialization in a high-dimensional space, rather than by passive survival due to insufficient confidence updates that would prevent the destruction process. While Mahalanobis distance may lose discriminative power in higher dimensional spaces, the observed seperation of state-action distributions enables kCELL to maintain multiple local experts corresponding to different dynamics. This results in stable prediction performance across successive dynamics changes.

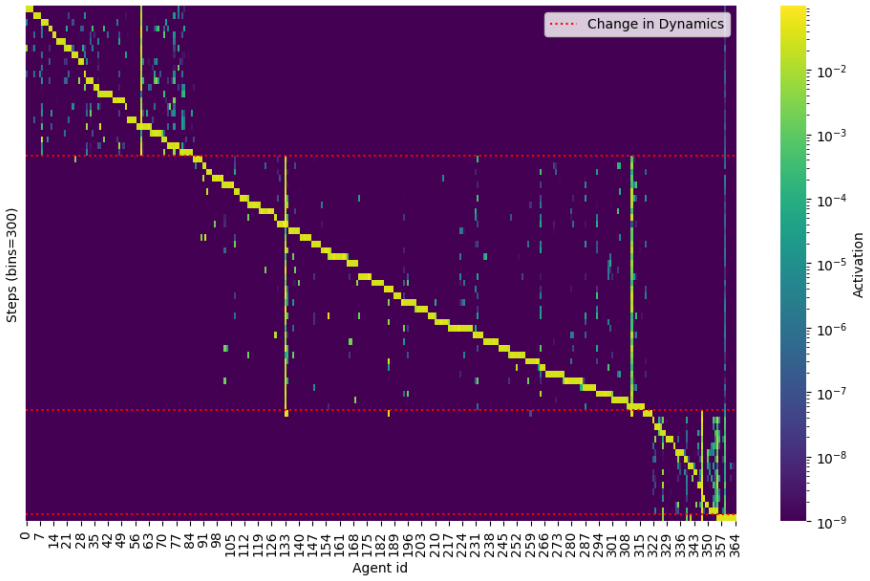


Figure 3.15: Heatmap of final Agents Activations on Training Datasets. y-axis correspond to the training data (in order) aggregated into bins of size 300 (from top to bottom) and x-axis corresponds to agents ids sorted from the oldest to the youngest (left to right). The dotted horizontal red lines corresponds to changes in dynamic i.e to a change of train dataset with a different value of g. The colors corresponds to activation levels of agents over each aggregated training data bins with yellow meaning high activation and blue meaning low activation.

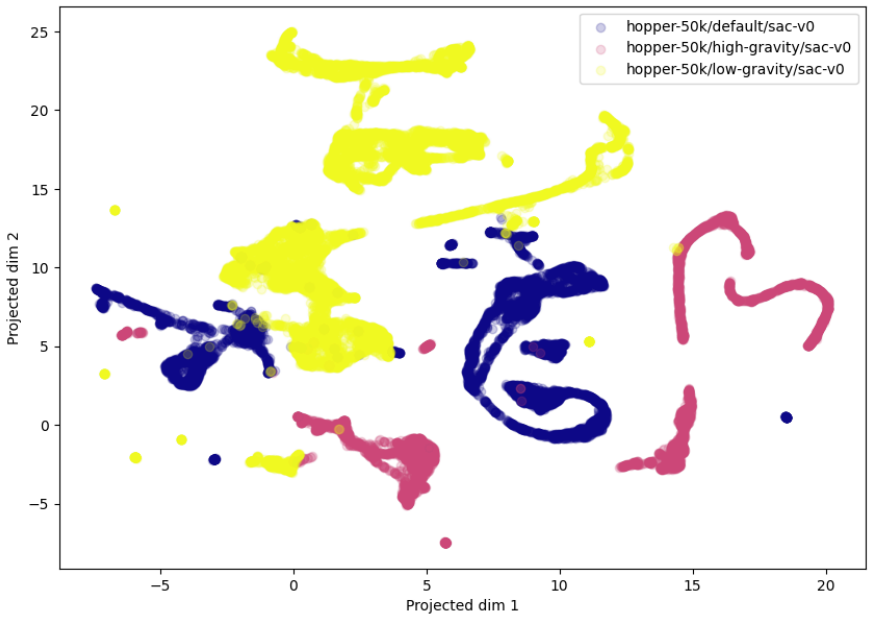


Figure 3.16: Visualization of UMAP projection of Training data where each point is colored depending on the training dataset it comes from (blue for $g = 9.81$; yellow for $g = 4$; pink for $g = 20$)

Discussion

The results presented suggest that kCELL were able to adapt to non-stationary dynamics through a combination of local specialization and structural plasticity with behaviors that depend strongly on the dimensionality and geometry of the feature space. In both environments, kCELL succesfully reduces prediction error on the currently observed dynamics demonstrating effective online adaptation without explicit task or concept drift detection modules. Compared to a Hoeffding Adaptive Tree baseline, kCELL achieves lower MAE for the low-dimensional InvertedPendulum environment when training and testing distributions coincide. It also achieves comparable performances for the higher-dimensional Hopper environment although with increased variability due to its structural adaptivity.

For the InvertedPendulum environment, changes in dynamics induced a rapid loss of relevance of agents trained on previous dynamics leading to their destruction or adjustment. This is reflected by sharp drops in agent activation, decreases in mean agent age, temporary surges in the number of agents followed by a stabilization and the observed increase in MAE on previously encountered dynamics. These indicate a population renewal process driven by confidence degradation and agent replacement. In this setting, partial forgetting emerges as a natural consequence of the overlap between the geometry of different dynamics and reflects appropriate structural adaptation rather than failure of retention.

In contrast, the Hopper environment exhibits persistent agent populations and stable prediction performance across successive dynamics changes. Although the number of agents grows monotonically, confidence-based destruction prevents the survival of agents that neg-

atively impact prediction accuracy. The absence of population turnover is therefore not attributable to ineffective agent pruning. Analysis of agent activations reveals a structured specialization pattern. The activation heatmap of the final agent population shows distinct blocks associated with different training phases. Older agents are selectively activated for earlier gravity regimes and younger agents for later ones. This organization indicates retention of regime-specific expertise rather than passive persistence of obsolete models. Indeed, the agents do not know when a new dynamics change might occur and which dynamics will replace the current one.

This interpretation is further supported by the UMAP analysis of the training data, which reveals clear separation between datasets generated with different gravity values. The existence fo well-separated regions in the state-action space provides a plausible explanation for the coexistence of multiple generations of agents without destructive interference. Thus, in this case, high dimensionality enables to allocate distinct subsets of agents to different dynamics. Although Mahalanobis distance may lose discriminative precision as dimensionality increases, for this experiment with 11-dimensional states and 3-dimensional actions (resulting in a 14-dimensional input vector), the geometric separation induced by gravity changes appears sufficient to maintain effective specialization.

Overall, the results suggest that kCELL implicitly adjusts the tradeoff between forgetting and retention based on the structure of the data distribution rather than relying on explicit task boundaries. In low-dimensional spaces, adaptation is achieved through agent replacement and partial forgetting, while in higher-dimensional settings with separable regimes, the system favors retention and specialization. These results highlight both the strength and current limitations of kCELL. It paves the way for future works on the use of more robust distance metrics and activation mechanisms to account for high-dimensional geometry.

Finally, we highlight the ability to analyze kCELL through the spatial organization and activation patterns of agents as a key advantage in terms of interpretability. Introspection of agent properties allows to identify which regions of the state-action space correspond to different dynamics. This provide understanding on how the model responds to non-stationary conditions which can be useful to debug, refine learning mechanisms or build fallback strategies. This relates to the properties we presented in section 3.1.4, showing that despite the design differences with oCELL, some related properties remain.

3.2.4 Online Stationary Modeling

3.2.5 Comparative Study of CELL Variants

3.2.6 Discussions and Limitations

Chapter 4

Solving Control Tasks with CELL

Building upon the Self Adaptive Context Learning (SACL) paradigm [6], we introduced CELL (Context Ensemble Local Learning) paradigm, which provides the core hypothesis and theoretical ground for designing spatialized multiagent learning systems. The algorithms derived from CELL, such as oCELL and kCELL, are online machine learning algorithms that exhibit unique explainability and interpretability properties useful for model introspection.

In this chapter, we leverage these properties to demonstrate the use of kCELL for solving control tasks without requiring prior knowledge of the system dynamics. Optimal Control tasks are often associated with constraints depending on legality, stability or safety requirements. In the field of data-driven control, enforcing the strict satisfaction of hard operational constraints remains a major challenge, as predictive models learned from data often lack the necessary robustness and verifiable guarantees required for safety-critical applications.

To address this, we show how kCELL can be used as the predictive model within a Model Predictive Control (MPC) scheme.

The core contributions presented in this chapter are threefold:

Integration with Solvers We demonstrate that kCELL's inherent local linearization enables a seamless integration with traditional non-linear optimization methods, such as Sequential Quadratic Programming (SQP) [7]. This makes a kCELL model directly exploitable by high-performance convex solvers for real-time constrained problems.

Enhanced Safety via Introspection The spatial organization of kCELL's local experts provides a knowledge map of model's training support. We leverage this map to dynamically control the optimization process's conservativeness, allowing the controller to safely manage exploration in poorly modeled states or prioritize stability in well-modeled regions.

Constraint Impact Analysis Building on this introspection capability, we propose a novel interpretability approach based on the Linear Quadratic Regulator (LQR) [28]. This method is used to analyze and quantify the precise impact of constraints on the optimization objective, establishing a baseline for unconstrained performance and providing a new layer of explanations.

This chapter is structured as following. Section 4.1 addresses the prerequisites for real-time control by presenting implementation guidelines focused on optimizing CELL algo-

rithms. Specifically it details how to achieve fast inference and learning through spatial indexing and GPU parallelization; and how to ensure differentiability of the core components using differentiable programming frameworks. In Section **??**, the integration of kCELL into a MPC scheme is described, establishing the foundational control. This leads to Section **??**, which focuses on safe control. It demonstrates the transition from soft constraints to robust hard constraints enforcement using kCELL within a SQP framework. Finally, Section 4.4 presents a novel explanation method based on LQR to justify the decisions in control regarding constraint enforcement by evaluating constraint impact on the optimization process.

4.1 Efficient kCELL for Real-Time Control

While kCELL provides a more robust and expressive instantiation of the CELL paradigm, capable of handling nonstationary online learning tasks and maintaining interpretability, its practical deployment in computationally demanding applications, such as model-based control, requires additional considerations. In particular, real-time control and optimization loops demand fast inference and the ability to compute gradients of the learned model with respect to inputs and control variables.

To meet these requirements, the CELL paradigm can be extended with differentiable representations and optimized implementations that exploit GPU parallelization and spatial indexing. This permits efficient evaluation of large populations of agents, allowing gradien-based optimization over the learned model. The following sections detail these practical strategies for making CELL implementations differentiable and computationally efficient, bridging the gap between expressive multiagent learning and real-time applications.

4.1.1 Efficient Implementation

4.1.2 MPC and Local Linearization

4.2 Exploratory MPC with kCELL

4.3 Conservative MPC with kCELL

4.4 Safe Explainable Control with Hard Constraints

Chapter 5

Scalable Non-Linear CELL

In chapter 3, we presented an ensemble learning algorithm to solve continuous supervised learning tasks. Then, in chapter 4, we demonstrated how to use our approach to continuously model the dynamics of a system in order to solve a constrained control task. Through our experiments, we have noticed that, when the state and action dimensions increased, the concept of neighborhood as we have defined it loses its consistency and informativeness due to the dilation of distances in the feature space .

Indeed, when the number of features increases, it is much rarer for an agent to be considered a neighbor of a new point. Therefore, the amoung of data required for training is much greater, and the number of agents created grows rapidly. Thus, we identify a need to limit the growth in the number of agents to increase sample efficiency and limit redundancy in the knowledge base. Until now, to mitigate this problem, we needed to rely on hard-to-tune hyperparameters to define the initial size of agents on each feature or on locality hypothesis on consecutive points among a given trajectory to identify relevant closest agents . In this chapter, we present SGP-CELL a novel approach based on Gaussian Processes [53] effectively tailored for scalable online learning. Our contributions are threefold:

- we propose a new spatialization approach for context agents based on Principal Component Analysis (PCA) [27] to robustify neighborhoods in larger feature spaces.
- we introduce a new learning process for individual agents based on model selection and greedy objective minimization.
- we demonstrate the performances and sample efficiency of SGP-CELL compared to a Sparse Gaussian Process baseline on a forward dynamics modeling task.

5.1 Related Works

Gaussian Processes (GP) are non-parametric Bayesian approaches to solve regression tasks while modeling uncertainty in predictions. GPs have been successful in robotics to model inverse or forward dynamics of a system to solve safe non linear control problems [3]. However, GP have a high computational cost with a learning complexity of $O\left(kN^3\right)$ with N the number of training points and k the number of optimization steps to find optimal kernel parameters,

which results from the inversion of the covariance matrix K. This makes GPs no able to handle large datasets.

Approximation methods like Sparse Gaussian Processes (SGP) alleviate this scaling issue. Instead of using the whole training dataset to build the model, a set of M inducing points (with $N \gg M$) are selected to represent the whole dataset. The inducing points allow for a cheaper low-rank representation for approximating the posterior distribution lowering the complexity to $O(NM^2)$ [48, 37].

Other works have extended SGP with Variational Inference to further enhance scalability with stochastic minibatch optimization to handle large datasets, improve generalization and reduce overfitting [49, 2].

For

5.1.1 Gaussian Process Regression

5.1.2 Online Gaussian Processes

5.2 SGP-CELL

5.2.1 Scaling Neighborhoods

5.2.2 Non-Linear Local Modeling

5.3 Experiments

5.4 Practical Design Guidelines

Based on our experience designing our multiagent systems based on context agents for supervised learning, we present in this section comprehensive overview of common obstacles that must be overcome for such a system to work properly, allowing the emergence of learning.

feedback (Δ)	update model	update shape		
Δ	✓			
		✓		
	✓	✓		

Table 5.1: kCELL features

5.5 Conclusion and limitations

Chapter 6

Speed Recommendation: Industrial Use Cases

The enforcement of the EU General Safety Regulation has accelerated the adoption of Intelligent Speed Assistance (ISA) systems in new vehicles, emphasizing the need for reliable embedded speed recommendations. Unlike classical speed control approaches that are centered on vehicle dynamics modeling, speed recommendation requires reasoning that considers the driver in the loop, introducing behavioral variability and acceptance constraints. Designing deployable systems further demands attention to safety compliance, homologation requirements, robustness under sensor failure and potential impacts on energy consumption. In this chapter, we discuss these challenges and outline design principles for building speed recommendation systems suitable for real-world deployment and propose search directions to advance the field toward operational intelligent speed recommendation solutions.

Chapter 7

Conclusion and Future Works

Bibliography

[1] M. Montaz Ali, Charoenchai Khompatraporn, and Zelda B. Zabinsky. A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. *J. Global Optim.*, 31(4):635–672, April 2005.

[2] Matthias Bauer, Mark Van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse Gaussian process approximations. *Advances in neural information processing systems*, 29, 2016.

[3] Felix Berkenkamp and Angela P Schoellig. Safe and robust learning control with Gaussian processes. In *2015 European Control Conference (ECC)*, pages 2496–2501. IEEE, 2015.

[4] Albert Bifet and Ricard Gavalda. Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pages 249–260. Springer, 2009.

[5] Clément Blanco-Volle, Nicolas Verstaevel, Stéphanie Combettes, Marie-Pierre Gleizes, and Michel Povlovitsch Seixas. Explainability and interpretability of an ensemble multi-agent system for supervised learning. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 335–350. Springer, 2024.

[6] Jérémy Boes, Julien Nigon, Nicolas Verstaevel, Marie-Pierre Gleizes, and Frédéric Migeon. The Self-Adaptive Context Learning Pattern: Overview and Proposal. In *SpringerLink*, pages 91–104. Springer, Cham, Switzerland, December 2015.

[7] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[8] Friedman Breiman. *Classification and Regression Trees*. Taylor & Francis, Andover, England, UK, October 2017.

[9] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[10] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[11] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[12] Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11):4948, 2021.

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):1–27, May 2011.

[14] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. *arXiv*, March 2016.

[15] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. *arXiv*, May 2018.

[16] Bruno Dato. *Apprentissage Permanent Par Feedback Endogène, Application à Un Système Robotique*. PhD thesis, Toulouse 3, 2021.

[17] Thomas G. Dietterich. Ensemble Methods in Machine Learning. In *SpringerLink*, pages 1–15. Springer, Berlin, Germany, December 2000.

[18] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *Ieee Access*, 6:28573–28593, 2018.

[19] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[20] Thibault Fourez, Nicolas Verstaevel, Frédéric Migeon, Frédéric Schettini, and Frederic Amblard. An ensemble Multi-Agent System for non-linear classification. *arXiv*, September 2022.

[21] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[22] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Stat.*, 29(5):1189–1232, October 2001.

[23] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE, 2018.

[24] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[25] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Mach. Learn.*, 110(3):457–506, March 2021.

[26] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *ArXiv*, abs/1308.4008, 2013.

[27] Ian Jolliffe. Principal component analysis. In *International Encyclopedia of Statistical Science*, pages 1094–1096. Springer, 2011.

[28] Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2):102–119, 1960.

[29] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 30, 2017.

[30] Andrei V Konstantinov and Lev V Utkin. Interpretable ensembles of hyper-rectangles as base models. *Neural Computing and Applications*, 35(29):21771–21795, 2023.

[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, May 2015.

[32] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 23(1):18, December 2020.

[33] David Lowe and D Broomhead. Multivariable functional interpolation and adaptive networks. *Complex systems*, 2(3):321–355, 1988.

[34] Octavio Loyola-González. Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View. *IEEE Access*, 7:154096–154113, October 2019.

[35] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[36] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[37] Andrew Naish-Guzman and Sean Holden. The generalized FITC approximation. *Advances in neural information processing systems*, 20, 2007.

[38] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2013.

[39] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *arXiv*, January 2012.

[40] Robi Polikar. Ensemble Learning. In *SpringerLink*, pages 1–34. Springer, New York, NY, New York, NY, USA, January 2012.

[41] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[42] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[43] Gregor Rohbogner, Simon Fey, Pascal Benoit, Christof Wittwer, and Andreas Christ. Design of a multiagent-based voltage control system in peer-to-peer networks for smart grids. *Energy Technology*, 2(1):107–120, 2014.

[44] Lior Rokach and Oded Maimon. Decision Trees. In *SpringerLink*, pages 165–192. Springer, Boston, MA, Boston, MA, USA, 2005.

[45] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *J. Math. Psychol.*, 85:1–16, August 2018.

[46] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[47] Shahaboddin Shamshirband, Nor Badrul Anuar, Miss Laiha Mat Kiah, and Ahmed Patel. An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique. *Engineering Applications of Artificial Intelligence*, 26(9):2105–2127, 2013.

[48] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.

[49] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574. PMLR, 2009.

[50] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[51] Sanford Weisberg. *Applied Linear Regression*. John Wiley & Sons, Ltd., Chichester, England, UK, January 2005.

[52] Barry Payne Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[53] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.

[54] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[55] Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

[56] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. *arXiv*, October 2020.