Contents

1 Introduction **3**
- 1.1 Background and Motivation . . . 4
- 1.2 Problem Statement . . . 4
- 1.3 Contributions . . . 4

2 State of the Art **5**
- 2.1 Speed Recommendation and Speed Control . . . 5
- 2.2 Online Machine Learning . . . 5
  - 2.2.1 Multi-Agent Learning . . . 5
  - 2.2.2 Ensemble Learning . . . 6
  - 2.2.3 XAI . . . 6
- 2.3 Optimal Control . . . 7
  - 2.3.1 Control with Learned Models . . . 7
- 2.4 Positioning . . . 7

3 Context Ensemble Local Learning (CELL) **8**
- 3.1 CELL Learning Paradigm . . . 9
- 3.2 oCELL: Multiagent Ensemble Learning with Orthotopes . . . 9
  - 3.2.1 Context Agents . . . 9
  - 3.2.2 Learning Rules . . . 12
  - 3.2.3 Comparative Study . . . 14
  - 3.2.4 Explainability and Interpretability . . . 16
  - 3.2.5 Limitations . . . 22
- 3.3 kCELL: Multiagent Ensemble Learning with Kernel Spatialization . . . 23
  - 3.3.1 Context Agents . . . 24
  - 3.3.2 Learning Rules . . . 26
  - 3.3.3 Adaptation to Non-Stationary Dynamics . . . 29
  - 3.3.4 Online Stationary Modeling . . . 37
  - 3.3.5 Comparative Study of CELL Variants . . . 37
  - 3.3.6 Discussions and Limitations . . . 37

4 Solving Control Tasks with CELL **38**
- 4.1 Efficient kCELL for Real-Time Control . . . 39
  - 4.1.1 Efficient Implementation . . . 40
  - 4.1.2 Differentiability and Local Linearization for Gradient-Based MPC . . . 45

4.2 Exploration MPC with kCELL . . . 48
4.2.1 Distance-based Competence . . . 49
4.2.2 Local Disagreement . . . 50
4.2.3 Comparative Study . . . 50
4.3 Conservative MPC with kCELL . . . 60
4.3.1 Distance-based Conservatism . . . 61
4.3.2 Comparative Study . . . 61
4.4 Towards Safe Explainable Control under Hard Constraints . . . 70

5 Scalable Non-Linear CELL 71
5.1 Related Works . . . 71
5.1.1 Gaussian Process Regression . . . 72
5.1.2 Online Gaussian Processes . . . 72
5.2 SGP-CELL . . . 72
5.2.1 Scaling Neighborhoods . . . 72
5.2.2 Non-Linear Local Modeling . . . 72
5.3 Experiments . . . 72
5.4 Practical Design Guidelines . . . 72
5.5 Conclusion and limitations . . . 72

6 Speed Recommendation: Industrial Use Cases 73

7 Conclusion and Future Works 74

Chapter 1

Introduction

1.1 Background and Motivation

1.2 Problem Statement

1.3 Contributions

3.2	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. [7]
3.2	Empirical Evaluation of oCELL Performance: A low-dimensional comparative study demonstrating oCELL's competitive performance against SotA ensemble algorithms. [7]
3.3	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 [25]. 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 [65] or electrical engineering, particularly with issues related to smart grids [59]. 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 [16].

In this chapter, we present various two instantiations of CELL, a MAS paradigm derived from the Self Adaptive Context Learning (SACL) [9] 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*) [16].

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 [54]. The objective is to encourage diversity among the models so that they do not all capture the same patterns in the data [24]. 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 [11] 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 [28], 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 [76] 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 [31, 26] and often used interchangeably. Thus, following the distinction made in [15], 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 [44]. 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 [47] and LIME builds an interpretable surrogate model that approximate a black-box model locally [58].

Within this discourse, machine learning models are typically categorized as white-box or black-box models [46]. **Black-box** models have opaque and complex inner workings that are difficult for an external observer to understand. This categorisation includes deep learning models [43] and some ensemble models [12, 54, 19, 40]. Conversely, **White-box** models have simpler and less opaque inner workings. These models, considered more explainable than black-box models, include linear regression models [73], decision trees [60] , 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 [20, 36]. **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

2.3.1 Control with Learned Models

Reinforcement Learning

Model-Based approaches

Exploration

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 [9, 27], 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 [9], 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 [9, 27], 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 3.2, oCELL is introduced, its capabilities are presented in terms of predictive performance and explainability, as well as its structural limitations. Section 3.3 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 CELL Learning Paradigm

3.2 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 [9] 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 3.2.1, the internal structure of context agents is described, 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 comparative study evaluating the performance of oCELL against other machine learning algorithms on a two-dimensional benchmark. Subsequently, section 3.2.4 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 3.2.5 discusses the limitations of oCELL and outline directions for future research, some of which are addressed in 3.3.

3.2.1 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 [27, 41]. These approaches typically partition the feature space into orthotopes and model the target function locally. For instance, [41], 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 [27], which uses the SACL paradigm [9] to progressively pave the feature space with context agents to address classification tasks. However, unlike [27], 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 n-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 [27], 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 [50].

To achieve this, oCELL allows multiple agents to contribute to the final prediction, in contrast to previous works [9, 22, 27], 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.

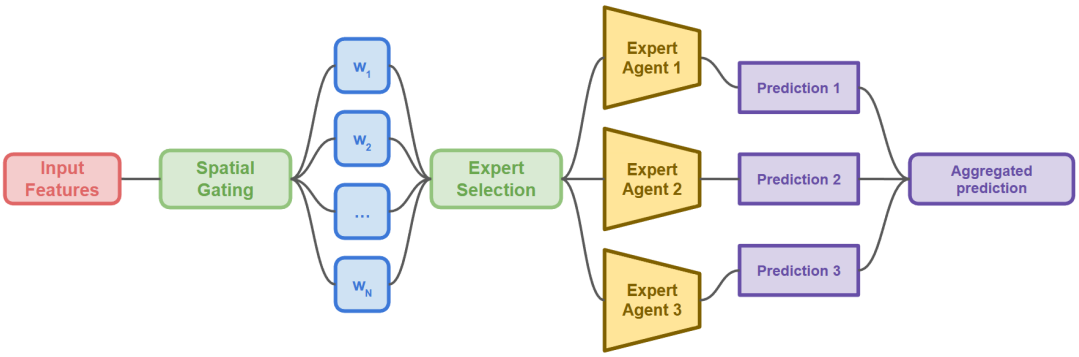


Figure 3.1: Illustration of the prediction process with agents in CELL paradigm

3.2.2 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.

3.2.3 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 [37]. This choice is motivated by the fact that state-of-the-art machine learning algorithms [19, 40] 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, 37]. 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.

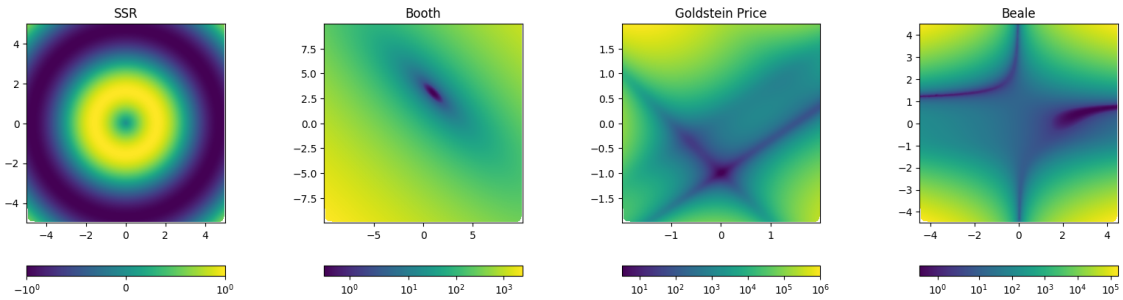


Figure 3.3: Visualization of the 2D functions used to generate the benchmark datasets. The color levels approaching yellow correspond to higher values, while the color levels approaching blue correspond to lower values.

We compare oCELL against widely used ensemble algorithms and common weak learners. The selected weak learners include decision trees [10], SVMs [18], and linear regression [73]. For ensemble methods, we consider both classical approaches such as gradient boosting [29] and random forests [12], alongside state-of-the-art algorithms: XGBoost [19] and LightGBM [40].

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 [53], xgboost [19], and lightgbm [40] 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 3.2.2, and *memory_ length* sepcifies the maximum memory size allocated to an agent.

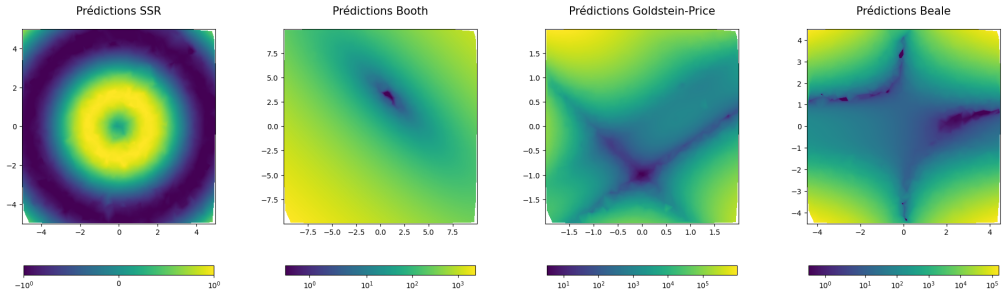


Figure 3.4: Visualization of oCELL predictions on data not included in the training dataset.

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.

3.2.4 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 [58], 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

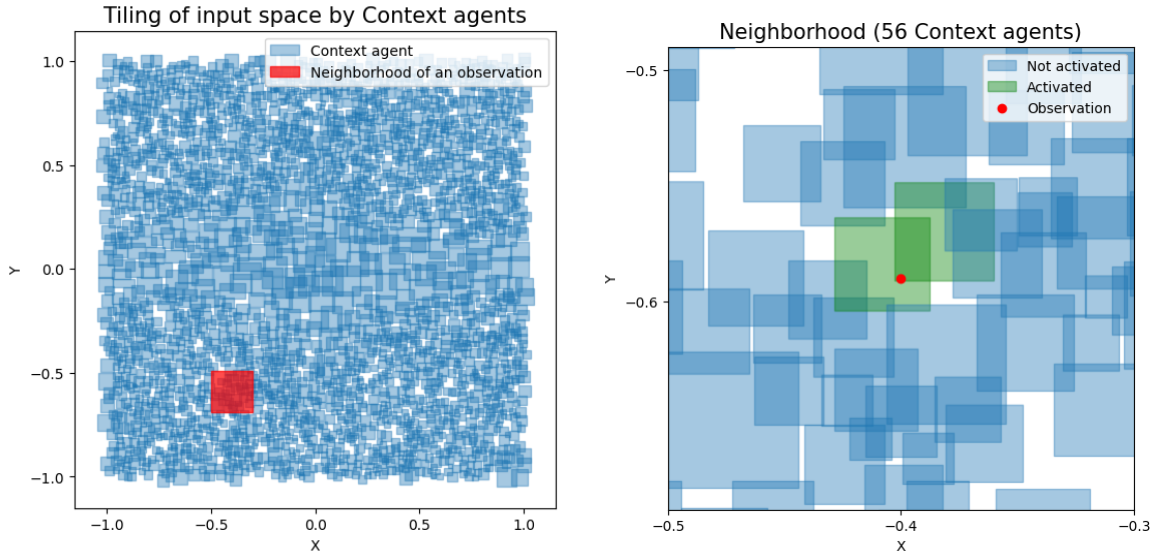
	hyperparameters	SSR	Booth	Goldstein-Price	Beale
oCELL	R	0.1	0.35	0.1	0.05
	τ_{good}	0.01	0.15	0.01	0.1
	τ_{bad}	0.2	0.2	0.2	0.3
	α	0.4	0.1	0.2	0.2
	memory_length	20	50	200	20
XG Boost	learning_rate	0.1	0.05	0.05	0.1
	max_depth	9	9	9	9
	n_estimators	300	300	300	300
	objective	abs. error	abs. error	squ. 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 [78, 37].



(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.

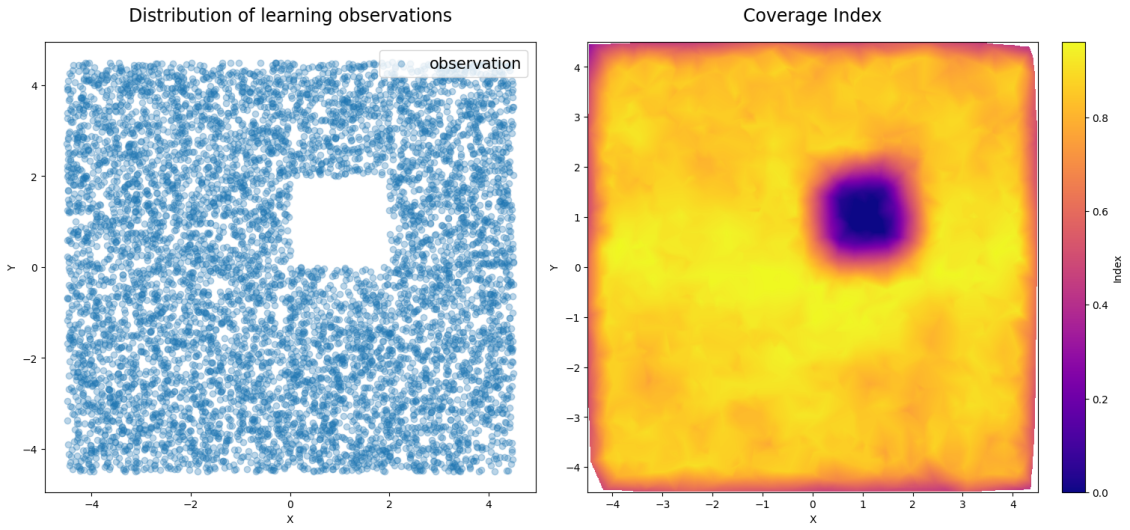


Figure 3.6: Comparison of the distribution map of observations used to train the model (left) and the visualization of the coverage index associated with this model (right).

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. 3.2.1). 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 [63]. 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.

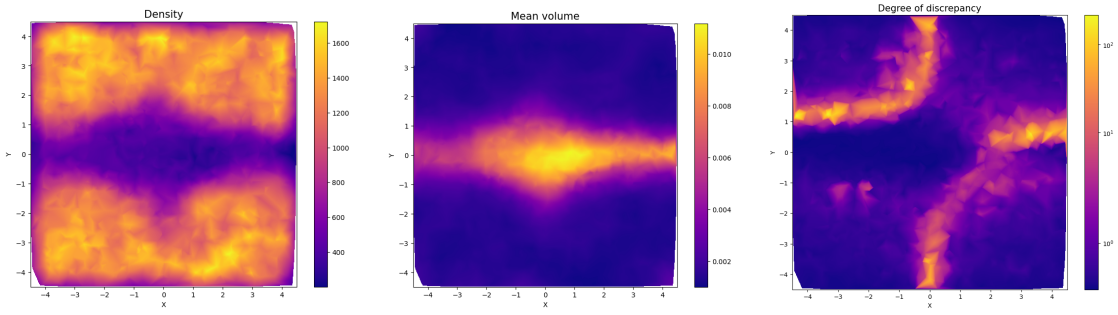
These two regions correspond to very different variation regimes of the Beale function. Indeed, the region containing the largest agents (low density) corresponds to a plateau where the gradient magnitude of the function varies little. In contrast, the region containing the smallest agents (higher density) 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

3.2.5 Limitations

We demonstrated that oCELL can address supervised learning tasks, but several limitations remain, which are tackled partly in section 3.3.

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.

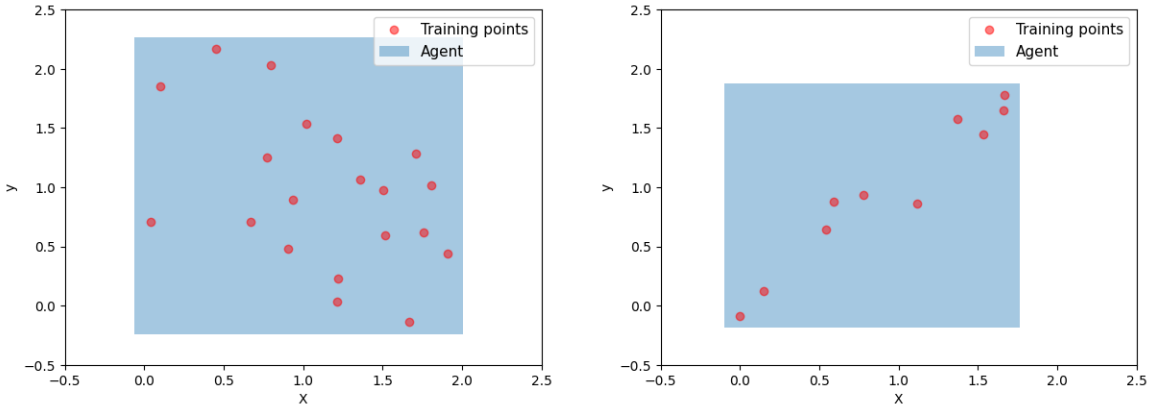
Furthermore, we observed that oCELL's agent destruction mechanism was inadequate. Agent overproduction was a frequently observed phenomenon, occasionally resulting in significant local redundancy and the persistence of harmful parasitic agents within the collective. This necessitates the development 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.

3.3 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 [45] and MoE (Mixture of Experts) [77] 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 [64].

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 3.3.1 describes the internal structure of context agents in kCELL, including their spatialization in the feature space and their prediction mechanisms. Then, Section 3.3.2 introduces the learning rules that cover agent creation, adaptation and self-organization leading to the emergence of learning in the system. Section 3.3.3 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 3.3.6 discusses the key differences between oCELL and kCELL and details the limitations of kCELL.

3.3.1 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:

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

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

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

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

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

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 [74] 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.

The notion of volume expressed in Section 3.2.1 (cf. Equation 3.2) still carries on. The volume of an agent is tied to the volume of a confidence ellipsoid defined by a theshold on squared mahalanobis distance. Thus the volume of an agent is proportional to the square root of the determinant of the associated covariance matrix

$$v\left(\mathcal{A}_i\right) \propto \sqrt{\det\left(\Sigma_i\right)}$$

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 $f\left(x\right) \in \mathbb{R}^m$ of the system is then given by

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

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

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

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

3.3.2 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 of the system $f\left(x\right) = \hat{y}$ 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.

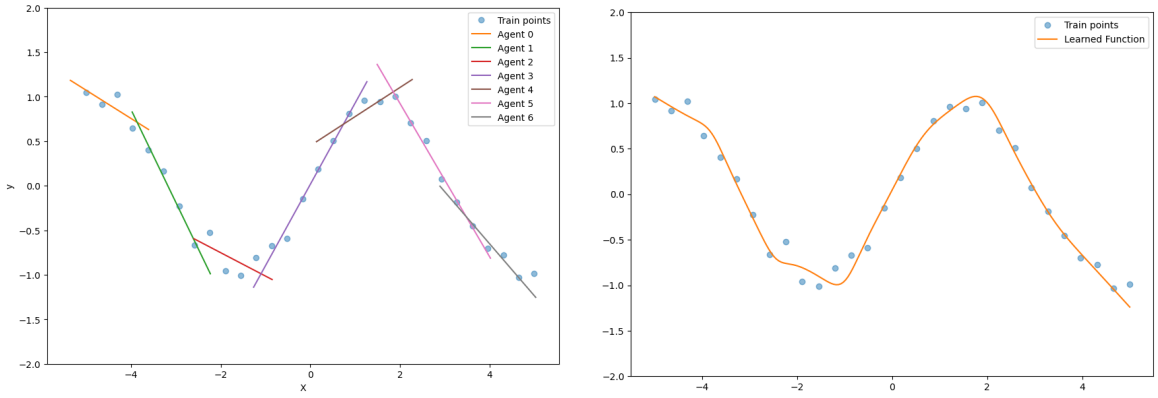
Finally, Table 3.5 presents a summary of the learning rules that describe the behavior of context agents in kCELL.

	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	$\varnothing$
	$C_{\mathcal{A}_i} \leq \tau_{\text{confidence}}$	Neighbor	destroy agent
$N > 1$	$\Delta_{\mathcal{A}_i} > 0$	Neighbor	update shape
	$\Delta_{\mathcal{A}_i} < 0$	Neighbor	update model
	$C_{\mathcal{A}_i} \leq \tau_{\text{confidence}}$	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 3.3.1.



(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.

3.3.3 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 [70] 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 [35] (with stable-baselines3 implementation [56]). 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 3.2.5). We also compare kCELL to Adaptive Hoeffding Trees [6] 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

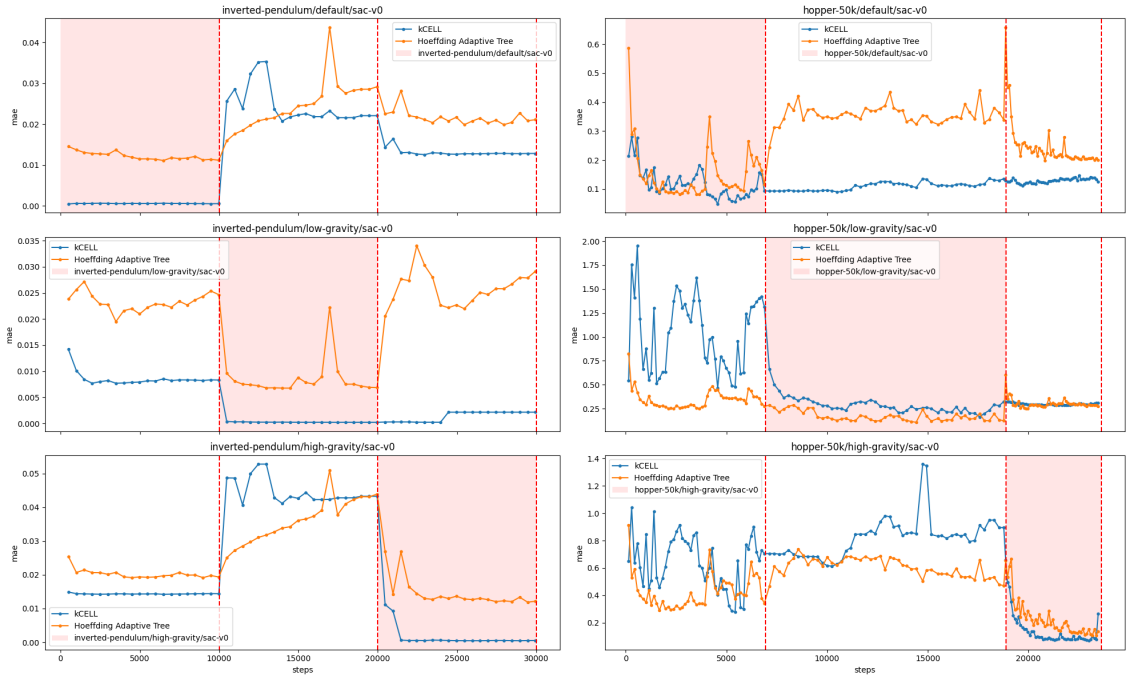
Table 3.6: Characteristics of datasets generated with different values of g. *State Size* stands for the number of features in states, *Nb Actions* stands for the number of continuous actions, *Dynamic Changes* corresponds the set of g values used to generate the datasets, *Nb Learning Steps* corresponds 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.



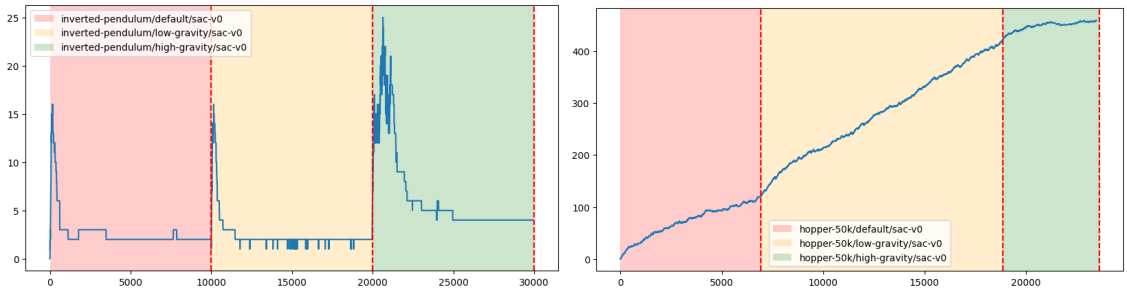
(a) InvertedPendulum (b) Hopper

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.



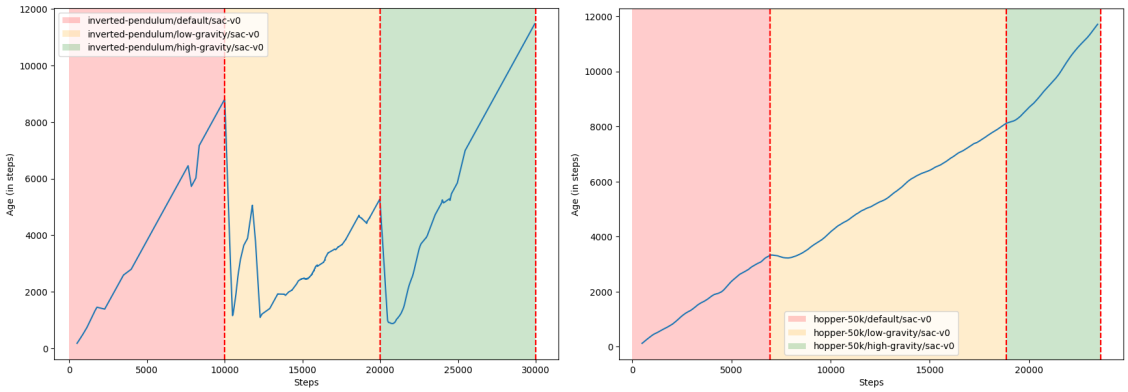
(a) InvertedPendulum (b) Hopper

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.



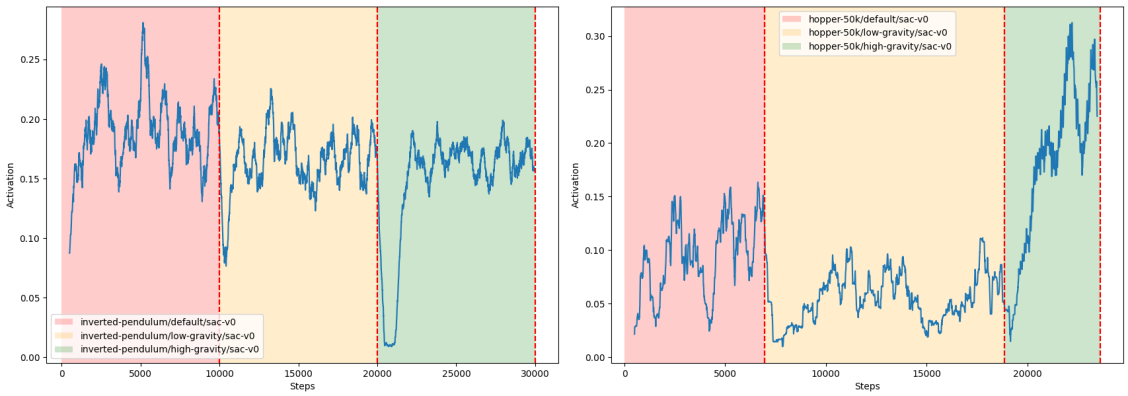
(a) InvertedPendulum (b) Hopper

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 [48] 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.

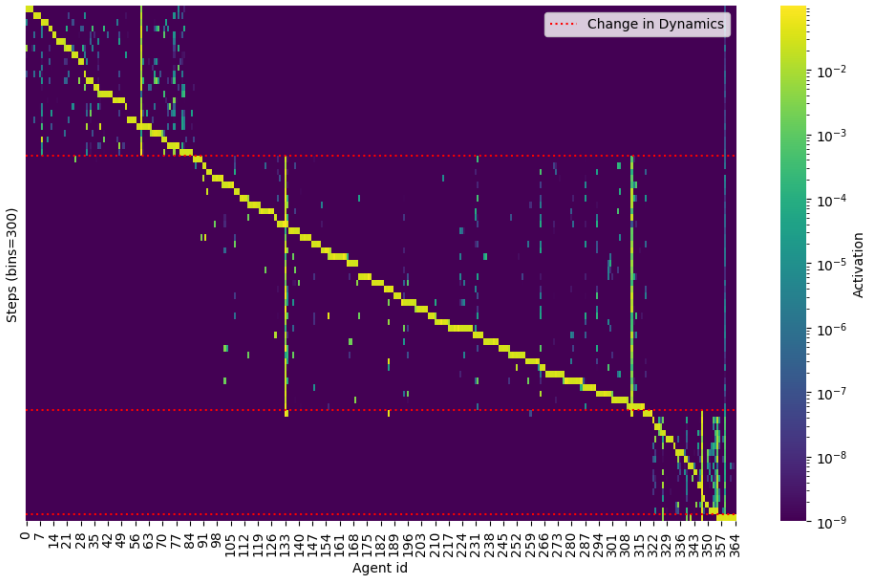


Figure 3.15: Heatmap of final Agents Activations on Training Datasets. y-axis correspond to the training data (in order) aggregated into bins of size 300 (from top to bottom) and x-axis corresponds to agents ids sorted from the oldest to the youngest (left to right). The dotted horizontal red lines corresponds 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.

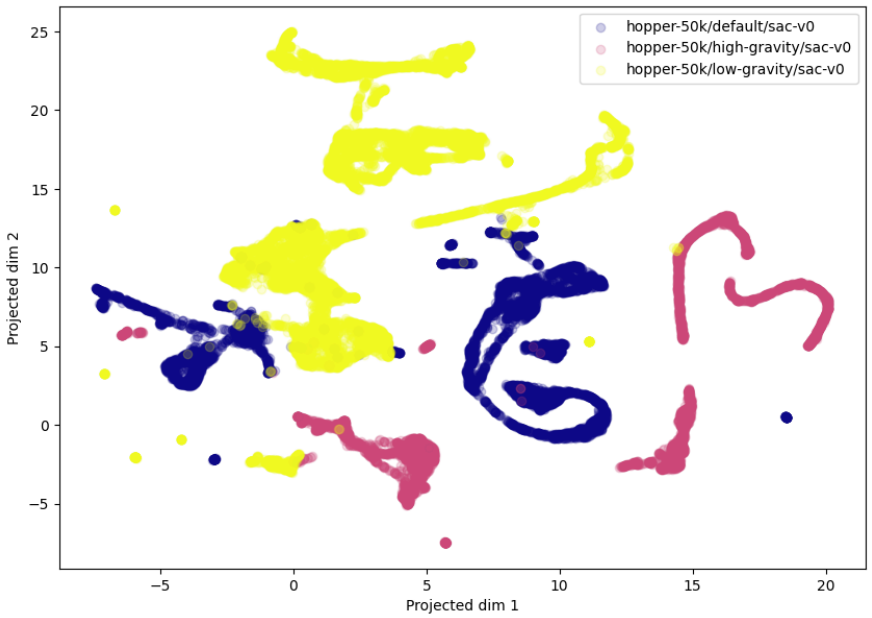


Figure 3.16: Visualization of UMAP projection of Training data where each point is colored depending on the training dataset it comes from (blue for $g = 9.81$; yellow for $g = 4$; pink for $g = 20$)

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 3.2.4, showing that despite the design differences with oCELL, some related properties remain.

3.3.4 Online Stationary Modeling

3.3.5 Comparative Study of CELL Variants

3.3.6 Discussions and Limitations

Chapter 4

Solving Control Tasks with CELL

Building upon the Self Adaptive Context Learning (SACL) paradigm [9], we introduced the CELL (Context Ensemble Local Learning) paradigm, which provides the core hypothesis and theoretical ground for designing spatialized multiagent learning systems. Within this paradigm, algorithms such as oCELL and kCELL rely on a population of agents spatialized in feature space. Each agent learns a local predictive model from a stream of data. These online learning systems exhibit intrinsic interpretability and explainability properties that arise from the spatial organization of agents, spatialized local learning and agent-level interactions.

This chapter investigates how these properties can be exploited for model-based control with learned model without prior knowledge of system dynamics. In this setting, control policies are built from the combination of a learned predictive model and an optimizer. This corresponds to the Model Predictive Control (MPC) framework, in which control actions are obtained by repeatedly optimizing predicted trajectories over a finite horizon [57].

When the predictions of learned models lie outside the regions in the state-action space that have been sufficiently represented in the training data, the model is inherently limited in terms of accuracy. This limitation needs to be considered in optimal control problems involving hard constraints related to safety, stability or physical feasibility [13]. Indeed, unreliable predictions lead to unreliable optimization results. While data-driven control methods have achieved promising results, they often rely on opaque inner workings and struggle to provide reliable guarantees when operating outside the support of the training data [55, 4].

In this chapter we show that kCELL's structural transparency provides introspection tools that can be directly embedded into control and optimization pipelines. kCELL exposes measurable quantities such as distance-based competence, local prediction disagreement that provide proxies for quantifying epistemic and aleatoric uncertainties (cf Section 2.2.3 for definitions). kCELL's properties can then be leveraged to guide exploration, regularize exploitation, and explain control decisions in a unified framework.

More specifically, the core contributions presented in this chapter are fourfold:

- We provide a discussion on how kCELL can be efficiently integrated into real-time, model-based control frameworks by exploiting its local linearization capabilities for gradient-based optimization and implementation strategies based on parallel computation or spatial indexing for fast trajectory unfolding in population-based MPC.

- We introduce introspection-driven strategies based on agent spatial organization and local disagreement for exploration of a low-dimensional environment without prior knowledge of the dynamics. We show that these strategies lead to more effective coverage of the state-action space than uninformed exploration mechanisms.
- We introduce an introspection-driven conservative regularization mechanism for trajectory optimization that leverages kCELL's explicit notion of local model competence, reducing the risk of extrapolation errors and improving robustness of control with learned dynamics.
- We present a novel local explanation method based on Linear Quadratic Regulator (LQR) [39] for constrained model-based control. This approach allows quantification of constraint impact and feature importance estimation in optimized control policies.

This chapter is organized as follows. Section 4.1 discusses the computational and differentiability properties of kCELL that make it suitable for real-time constrained control within a MPC scheme. Section 4.2 focuses on exploration of unknown dynamics and shows how the spatial organization and local interactions of agents can be exploited to drive data-efficient exploration. Section 4.3 addresses robust exploitation by introducing knowledge-aware regularization for trajectory optimization with kCELL. Finally, Section 4.4 introduces a novel LQR-based explanation mechanism for constrained controllers to evaluate the impact of constraints on the optimization process and to estimate feature importance, highlighting how kCELL allows to obtain interpretable and explainable control decisions.

4.1 Efficient kCELL for Real-Time Control

kCELL provides a more robust and expressive instantiation of the CELL paradigm. It is capable of handling non-stationary online learning tasks and maintaining interpretability. However, its practical deployment in model-based control requires additional computational and structural considerations.

Model Predictive Control (MPC) relies on repeated model evaluations within an optimization loop. When using learned models, inference speed and mathematical structure are important to the feasibility and reliability of control, especially if real-time constraints need to be met.

To this end, two main requirements must be addressed. First, population-based optimization methods commonly employed in data-driven MPC, such as the Cross-Entropy Method (CEM) [23], require fast and batched predictions over large sets of candidate trajectories whose computational cost must remain predictable across control steps. Purely sequential inference with naive expert lookup in kCELL prohibitively expensive. Second, gradient-based optimization methods, such as Sequential Quadratic Programming (SQP) [57], require differentiable predictive models in order to compute local linearization of the dynamics and to handle hard constraints efficiently.

kCELL must therefore comprise optimized implementations to enable fast batched inference and differentiable representations to support gradient-based control methods. Regarding computational efficiency, in kCELL, expert selection can be accelerated significantly

through spatial indexing or parallel execution, enabling efficient batched inference on modern hardware such as GPUs. Regarding differentiability, when activation functions and internal models are differentiable, kCELL admits analytic gradients and straightforward local linearization, making it naturally compatible with gradient-based MPC formulations.

Section 4.1.1 discusses strategies that can be implemented to accelerate inference and learning in kCELL, making it suitable for use with random shooting optimization methods like CEM. Then, Section 4.1.2 shows how the local structure of kCELL lead to trivial local linearizations of the learned dynamics, making kCELL suited for constrained MPC solved using SQP.

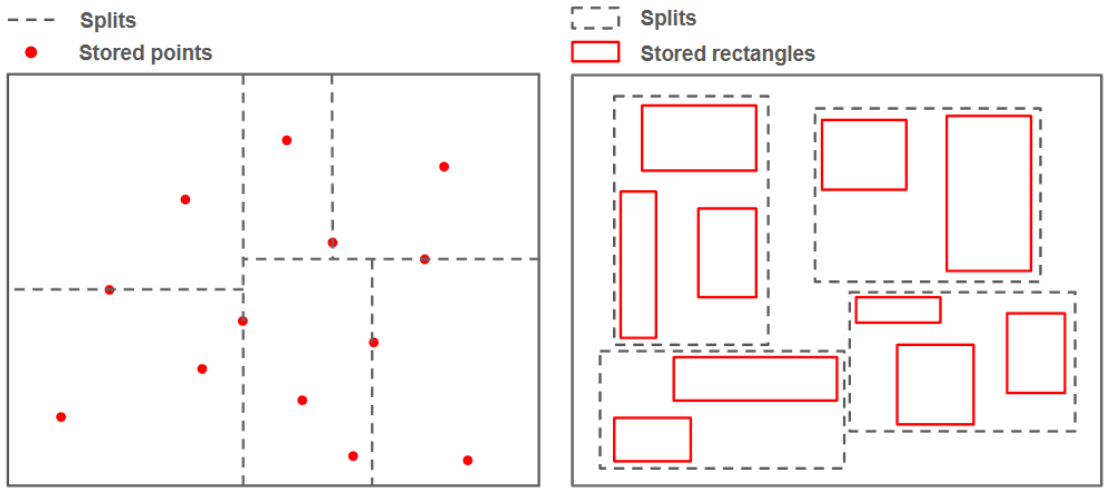
4.1.1 Efficient Implementation

Population based trajectory optimization algorithms are broadly used in robotics for their robustness to non-convex and non-smooth dynamics or cost functions [17]. This includes stochastic shooting methods such as Model Path Predictive Integral (MPPI) or CEM (which is used in Section 4.2). These methods rely heavily on parallel evaluation of large population of candidate trajectories at each control step within the MPC framework. In this section, we show that GPU parallel brute force is more reliable than geospatial indexing for nearest neighbor selection to speed up learning and inference process in kCELL for population based MPC.

Geospatial Tree Indexing

In CELL instanciations like kCELL and oCELL, a learning step always begins with a neighboring agent selection phase. Since the agents are spatially distributed in the feature space, then the selection of neighboring agents can be broken down into two components: distance calculation to find the closest agents and thresholding of those distances to discriminate neighbors.

Because agents are spatially distributed within the feature space, managing the ensemble of agents as a geospatial database can be considered. In this case, to speed up nearest neighbor searches, spatial indexing techniques can be leveraged to select the k-closest agents with a reduced complexity. Indexing in geospatial databases is most commonly performances using tree-based methods such as KD-tree or R-tree [33]. The KD-tree (for k-dimensional tree) is a binary tree data structure. It recursively partition space by splitting along alternating dimensions, making it efficient for range queries and nearest neighbor searches on point data [3] (cf Figure 4.1a). The R-tree is a hierarchical tree structure designed for indexing multi-dimensional objects like rectangles or polygons. Each node of the tree represents a bounding rectangle that encloses child nodes. R-trees are particularly suited for range, intersection and nearest neighbor queries in GIS and geospatial databases [34] (cf Figure 4.1b).



(a) KD-tree (b) R-tree

Figure 4.1: Illustration of KD-tree and R-tree on 2D data

In the context of CELL instantiations, R-trees are better suited than KD-trees because they are designed to handle polygons and orthotopes, i.e n-dimensional generalizations of rectangles. Therefore, agents can be indexed by their bounding orthotope. For kCELL, this is the orthotope encompassing the agent's confidence interval specified as a parameter. Furthermore, R-trees can be incrementally updated, making them compatible with the online learning nature of CELL instantiations [34]. Finally, with a spatial index based on a R-tree, the selection process can be broken down into three ordered steps: searching for the k-closest agents in the R-tree, calculating the distances between the point and the k-closest agents, and finally thresholding the distances to discriminate the neighbor agents.

Implementing a R-tree as a geospatial index over the agents reduces the neighborhood searches from $O(n)$ in the naive case to $O(\log n)$. However, for insertion and deletion, the average time complexity increases from $O(1)$ in the naive case to $O(\log n)$ with geospatial indexing. While geospatial indexing seems less advantageous in terms of insertion and deletion, it should be remembered that agent creations and destructions are generally less frequent then neighborhood searches in kCELL.

GPU Parallelization

However, as the dimensionality of the feature space increases, the structural advantages of the R-tree diminish due to distance distention and bounding box overlap making the search complexity degenerate to $O(n)$. In such cases, the overhead induced by geospatial indexing offers no advantage over a brute-force approach. To address these limitations, the embarrassingly parallelizable nature of neighborhood search in kCELL (and oCELL) can be exploited through GPU acceleration. By shifting from pointer-based tree structures to dense tensor representations, kCELL (and oCELL) can leverage massive parallel throughput as the selection phase can be executed as a vectorized linear algebra operation.

Memory Layout To maximize the efficiency of using the GPU for agent selection, agent data is stored in contiguous memory block. For example, for kCELL system, we define two global agent tensors for storing the parameters of the activation functions,

$$\mathcal{T}_\mu \in \mathbb{R}^{N\times n},\ \mathcal{T}_\Sigma \in \mathbb{R}^{N\times n\times n}$$

where N is the maximum agent capacity of the agent pool and n is the number of features. $\mathcal{T}_\mu$ is for storing the means and $\mathcal{T}_\Sigma$ is for storing the covariance matrices. With this layout, the GPU hardware can easily saturate its bandwith for efficient distance computations.

Dynamic Agent Population In CELL systems, the number of agents is dynamic. Agents can be created or destroyed depending on the needs of the system. In a GPU parallelization context with contiguous memory storage, managing a dynamic set of agents requires efficient creation and deletion strategies to avoid costly memory allocations.

The most straightforward approach consists in a dynamic reallocation of memory each time an agent is created or destroyed as illustrated in Figure 4.2. Although easy to implement, this approach is unsuitable for high-frequency learning loops due to the overhead induced by re-allocating GPU memory and copying of existing data.

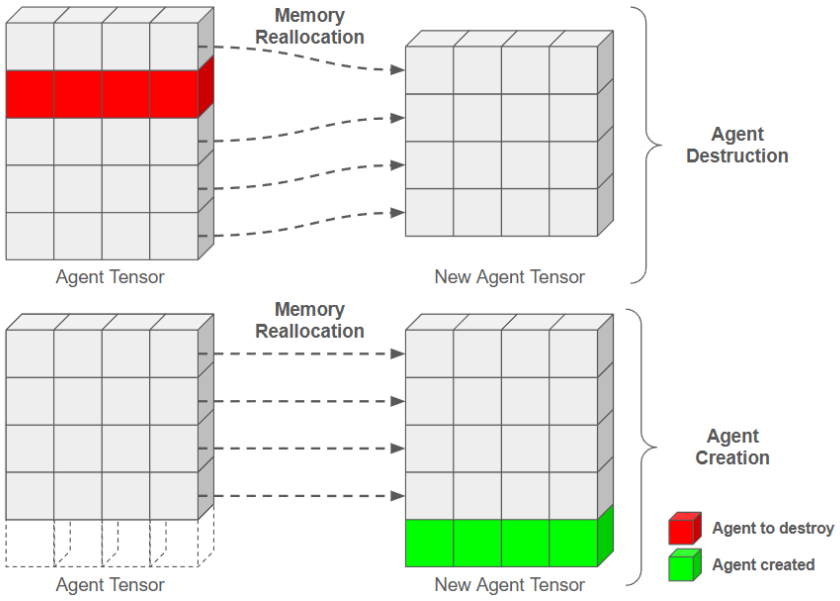


Figure 4.2: Illustration of agent creation and destruction in dynamic reallocation regime

To achieve $O(1)$ updated complexity, another strategy can be considered. Fixed-capacity tensors are pre-allocated to store agents data. A boolean masking vector $\mathcal{M} \in \{0,1\}^N$ is maintained to track which memory slot contains an agent or not.

Agent creation then consists in assigning new agents data to the first index i such that $\mathcal{M}_i = 0$. If the agent population exceeds the predefined capacity, standard dynamic resizing strategies can be employed.

Conversely, agent deletion sets $\mathcal{M}_i = 0$, marking the slot as available to host data of future agents as illustrated in Figure 4.3. This effectively hide the agent from the selection kernel without moving data.

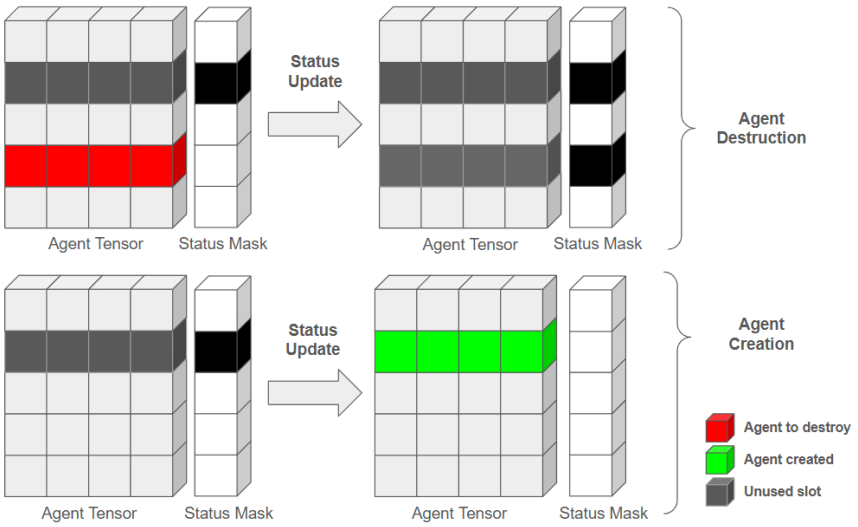


Figure 4.3: Illustration of agent creation and destruction in preallocated memory pool with masking regime

For selection, the mask $\mathcal{M}$ is used to filter the active memory slots and retrieve only relevant distance computations (i.e those corresponding to existing agents).

Comparison between GPU and Tree Selection

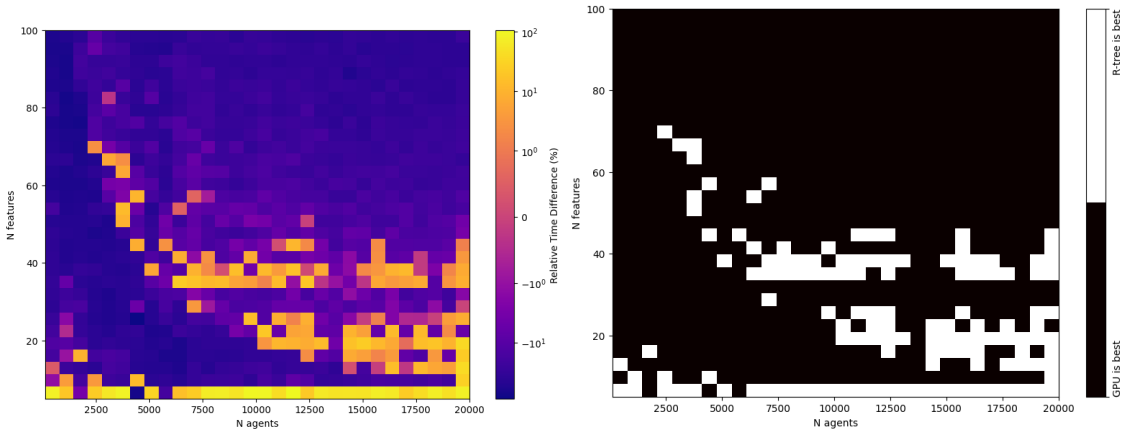
The choice of using a geospatial index or the parallelization enabled by a GPU depends on the number of features and the quantity of agents required for learning the function underlying the data which is highly correlated to its compexity.

To evaluate the impact of this structural shift, we benchmark neighborhood selection in Python using *PyTorch* [51] for GPU accelerated computations and using *Rtree* package [30] for efficient R-tree implementation. We compare the two selection approaches presented previously on a synthetic nearest neighbor selection task for variable amount of features and agents. The synthetic centroids are generated to form 100 clusters to simulate high density manifolds. As shown in Chapter 3, in CELL instantiations like oCELL and kCELL, agents organize spatially according to the data encountered. In real-world cases, these data points are rarely uniformly distributed. Indeed, uniform distribution could overestimate the volume of search. Using clustered centroid distribution allows a fairer comparison for R-trees whose main strength is to prune empty space.

From a set of 1000 query points, the time to find the 5-nearest neighbors is measured and averaged across 3 cycles. Figure 3.15 exposes a heatmap of the relative execution time difference in percentage expressed as

$$\Delta_t = \frac{\Delta_{\text{GPU}} - \Delta_{\text{R-tree}}}{\Delta_{\text{R-tree}}} \times 100$$

where Δ_{GPU} and $\Delta_{\text{R-tree}}$ the execution time of a nearest neighbor query averaged over 3 cycles. Figure 4.4b shows which strategy has obtained lowest execution time in each studied configuration.



(a) Relative Execution Time Difference Heatmap (b) Winner Binary Heatmap

Figure 4.4: Comparison of the execution times of GPU and R-tree on various $\{n, N\}$ configurations for a clustered distribution of synthetic centroids. contains a heatmap of the relative execution time difference between the two strategies (Δ_t). Shades of yellow correspond to $\Delta_t > 0$, i.e to R-tree being faster. Conversely, shades of blue correspond to $\Delta_t < 0$, i.e to GPU being faster. 4.4a contains a binary heatmap clearly showing in which configuration which strategy was faster. White cells corresponds to configurations in which R-tree was faster and black cells correspond to configurations in which GPU was faster.

The benchmarking results show that R-tree maintains a competitive edge in low-dimensional regimes ($n \leq 5$). This is an expected result as R-trees struggles in higher dimensions due to the curse of dimensionality. In higher dimensional regimes the distinction between nearest and farthest points becomes meaningless and pruning becomes ineffective [42].

However, the performances of the R-tree compared to GPU does not seem monotonic. In Figures 4.4a and 4.4b, we notice a non linear phase transition area (looking like a "semi-banana") that extends from $N = 2500$ centroids and $n = 70$ features to $N = 20000$ and $n = 50$. Within this region, R-tree is competitive or faster than GPU.

As shown in Figure 4.5, the competitive advantage of R-trees within the non linear phase transition area fades when the centroids are uniformly distributed. In this case the R-trees are almost never faster than GPU computation. It shows that the structure of data needs to be analyzed beforehand to make sure R-tree indexing is a pertinent indexing option. This is an expected result because R-trees are particularly adapted to prune large empty spaces. The uniform distribution of centroids does not exhibit exploitable structures for the R-tree. This is not an issue for the GPU based nearest neighbor computations that is a brute force approach that does not rely on structure.

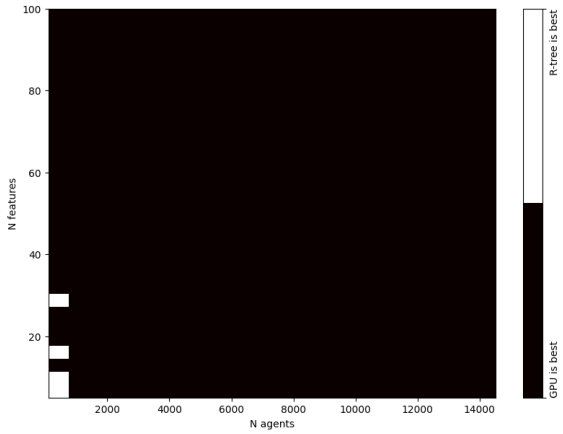


Figure 4.5: Winner binary heatmap with uniformly distributed synthetic centroids

These observations indicate that the performance of nearest neighbor selection with R-trees compared to using GPUs is better in low dimensional regimes ($n \leq 5$) and depends heavily on the underlying data structures when the number of dimensions increases ($n \geq 5$) due to the curse of dimensionality.

From an engineering standpoint, GPU-based selection provides a more predictable and stable latency profile, independent of data distribution. While R-tree indexing can provide large gains in very specific sparse regimes, its performance is highly sensitive to the latent structure of the learned model (i.e the distribution of agents).

In an online learning setting for adaptive control in a MPC scheme without prior knowledge of the dynamics of the environment, the data distribution is not known in advance and may evolve over time. For this reason, the GPU-based brute-force selection is preferred for its *deterministic latency* and *hardware scalability*. Indeed, GPU execution time scales predictably with n and N, whereas R-tree latency exhibits high variance depending on tree depth and splits overlap. Moreover, GPUs naturally handles batch queries as a single high-throughput operation, which can be deemed useful for population-based optimization.

The implementation strategies discussed in this section address the primary computational bottleneck of kCELL for use in a MPC scheme, namely the repeated selection and evaluation of local experts under strict real-time constraints. By leveraging either spatial indexing or GPU brute force selection, kCELL can be integrated into population-based trajectory optimizers while maintaining predictable inference and learning latency.

However, computational efficiency alone is not sufficient for kCELL to be deployed with gradient-based optimizers that could potentially handle hard constraints. Such approaches require predictive models to admit local linearizations.

4.1.2 Differentiability and Local Linearization for Gradient-Based MPC

Fast and scalable inference in kCELL is essential for its integration into real-time MPC frameworks, especially when using population-based or gradient-based trajectory optimization methods that rely on repeated model evaluations.

However, computational efficiency alone is insufficient when MPC formulation rely on gradient-based optimization methods that exploit local structures to enforce hard constraints. Such approaches require predictive models that admit reliable local linearizations of the learned system dynamics.

In this section, we show that kCELL naturally provides local affine approximations of the dynamics that are directly compatible with gradient-based optimization approaches such as Sequential Quadratic Programming (SQP).

Differentiability in CELL instantiations

In CELL instantiations, online learning is performed by maintaining a set of self-organizing spatialized agents that map the target function locally (cf Chapter 3). A context agent is denoted as

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

where $\phi_i : \mathbb{R}^n \mapsto]0, 1]$ is an activation function that maps features to a level of expertise for predicting a given input, and $f_i : \mathbb{R}^n \mapsto \mathbb{R}^m$ is local internal prediction function that maps features to target outputs.

In CELL instantiations, the global prediction is obtained by aggregating the predictions of agents, leveraging activation and prediction functions through an aggregation function. If the activation function, the prediction function and the aggregation function are differentiable, then, by standard composition rule, the global prediction function of the system is differentiable.

For kCELL, we consider a RBF kernel as the activation function, an affine function as the prediction function and a weighted average as the aggregation function. The final prediction is obtained by

$$f(x) = \frac{\sum_i \phi_i(x) f_i(x)}{\sum_i \phi_i(x)} = \sum_i w_i(x) f_i(x) \tag{4.1}$$

where $w_i(x)$ are the normalized weights such as $\sum_i w_i(x) = 1$.

As all the involved functions are differentiable with respect to x, the derivative of the prediction function of the system if given by

$$\frac{\partial f}{\partial x}(x) = \sum_i \left(\frac{\partial w_i}{\partial x}(x) f_i(x) + w_i(x) \frac{\partial f_i}{\partial x}(x) \right) \tag{4.2}$$

This property makes kCELL usable in gradient-based nonlinear optimization frameworks. However, we argue that a principled approach to linearization can be leveraged by exploiting kCELL's internal structure. In typical model-based control, non-linear dynamics are lienarized via a first-order Taylor expansion to fit the requirements of convex optimization [57]. However, when the predictive model is already an ensemble of local linear structures, applying a Taylor expansion becomes conceptually redundant. For this reason, we propose to bypass standard differentiation in favor of the structural local linearization capabilities of kCELL.

Local Linearization

In kCELL, the spatial organization of agents and their linear internal models allows to extract local affine models without computing full Jacobians or Taylor expansions.

In MPC, the predictive model predicts the next state s_{t+1} from $x_t = [s_t, u_t]$ where s_t is the current state of the system and u_t is the action to take. Assuming agents have linear internal models $f_i(x_t) = A_i s_t + B_i u_t + c_i$, the local affine model around a query point x_t is given by

$$
\begin{aligned}
A(x_t)\, s + B(x_t)\, u + c(x_t) &= \sum_i w_i(x_t)\, f_i(x_t) && (4.3)\\
&= \sum_i w_i(x_t)\, (A_i s_t + B_i u_t + c_i) && (4.4)\\
&= \underbrace{\sum_i w_i(x_t)\, A_i}_{A(x_t)}\, s_t + \underbrace{\sum_i w_i(x_t)\, B_i}_{B(x_t)}\, u_t + \underbrace{\sum_i w_i(x_t)\, c_i}_{c(x_t)} && (4.5)
\end{aligned}
$$

This approach offers a significant advanatge for control because it preserves the dynamics as learned by the model in the first place.

Sequential Quadratic Programming (SQP)

Sequential Quadratic Programming (SQP) is a local optimization method designed for solving nonlinear constrained problems. The optimization problem to solve is a nonlinear program formulated as

$$
\begin{aligned}
\min_{u_{0:T-1}} \quad & J(s_{0:T}, u_{0:T-1})\\
\text{s.t.} \quad & s_{t+1} = f(s_t, u_t)\\
& h(s_t, u_t) \leq 0\\
& g(s_t, u_t) = 0
\end{aligned}
$$

where $J : \mathbb{R}^{n \times T} \mapsto \mathbb{R}$ is the cost function, f the dynamic function describing the system, h and g correspond respectively to the supplementary inequality and equality constraints.

At each iteration, the original nonlinear program is approximated by a Quadratic Program (QP) in which the objective is quadratic and the dynamics and constraints are linearized around a reference trajectory

$$
\begin{aligned}
\min_{u_{0:T-1}} \quad & \sum_{t=0}^{T-1} \left(s_t^\top Q_t s_t + q_t^\top s_t + u_t^\top R_t u_t + r_t^\top u_t \right)\\
\text{s.t.} \quad & s_{t+1} = A_t s_t + B_t u_t + c_t\\
& h_{\text{lin}}(s_t, u_t) \leq 0\\
& g_{\text{lin}}(s_t, u_t) = 0
\end{aligned}
$$

where A_t, B_t and c_t define the locally linearized dynamics obtained from a first-order Taylor expansion, h_{lin} and g_{lin} correspond respectively to the linearized inequality and equality

constraints obtained from a first-order Taylor expansion, the matrices Q_t, R_t and vectors q_t, r_t are obtained from a second-order Taylor expansion of the original cost J around the reference trajectory τ_t.

Solving this QP yields a new trajectory τ_{t+1} than is used to compute a descent direction $\tau_{t+1} - \tau_t$. Since both SQP and kCELL rely on locally valid approximations of the dynamics, the update step must be restricted to remain within the region where the linearization is accurate. To this end, a line search [32] or trust-region strategy [21, 62] is typically employed to determine an appropriate step size that keeps linearization error contained and ensures sufficient cost decrease. This mechanism guarantees stable convergence when optimization trajectories with learned dynamics. The process is repeated until convergence.

Due to its spatialized agent-based structure, kCELL provides naturally the linearized dynamics needed for the local QP. As shown in Equation 4.5, for a query point $x_t = [s_t, u_t]$, the local affine model

$$s_{t+1} \approx \underbrace{A\left(x_t\right)}_{A_t} s_t + \underbrace{B\left(x_t\right)}_{B_t} u_t + \underbrace{c\left(x_t\right)}_{c_t}$$

provides exactly the matrices needed for the linearized dynamics in the QP. This way the approximated QP subproblem is fully compliant with the dynamics initially learned by the model, without having to build a proxy through differentiation and Taylor expansion. Consequently, SQP-based MPC provides a principled and efficient way to exploit kCELL's local structure while being able to enforce hard constraints.

In this section we discussed about the integration of kCELL into real-time MPC pipelines by exploiting both its computational and structural properties. First the differentiable and modular nature of kCELL enables highly parallel trajectory unfolding, making population-based optimizer tractable through GPU based implmentations. This approach is particularly adapted for large-scale trajectory sampling and global solution exploration. Then, we showed that kCELL's agent-based structure provides both differentiability and explicit local affine models of the learned dynamicsk for gradientt-based MPC. This property naturally allow the use of SQP, which exploits local linear models to efficiently enforce hard constraints and produce smooth control trajectories. Ultimately, we showed that using a kCELL predictive dynamics model, both population-based and gradient-based optimization schemes could be leveraged.

Beyond computational efficiency, kCELL exposes introspection signals such as distance-based competence and local model disagreement that quantify the reliability of the predictions along candidate trajectories. in the remainder of this chapter, we explore the use of these introspection signals to improve control in terms of exploration and exploitation under operational constraints.

4.2 Exploration MPC with kCELL

The integration of kCELL into SQP framework provides a solid and efficient way of exploiting current model knowledge. However, the performances of any model-based controller is bounded by the quality of the learned dynamics. In settings where no prior knowledge of the system is available, the controller must balance the minization of the task cost with the need to collect informative data in unknownn or poorly modeled regions of the state-action space.

Conventional exploration methods typically depend on heuristic noise processes (e.g Ornstein-Uhlenbeck noise [72]) or on uncertainty-based mechanisms derived from auxiliary models, such bayesian posterior variances in Gaussian Processes [67, 61], intrinsic motivation signals in Curiosity-driven exploration [52], or novelty scores learned through Random Network Distillation [14]. While effective in many situations, these approaches quantify uncertainty through probabilistic inference or learned proxies that are determined by model design choices, whereas kCELL exposes uncertainty as a structural and geometric property of its spatialized local learning architecture.

kCELL's spatialized agent-based architecture provides two distinct introspection signals that can be directly embedded into the MPC objective function to drive exploration of unknown states: distance-based competence and local model disagreement.

Distance-based competence quantifies how far a state-action pair lies from regions that are sufficiently covered by agents, providing a geometrical grounded measure of feature space epistemic uncertainty. Local model disagreement captures variability in the predicted dynamics across neighboring agents, yielding a local estimate of predictive uncertainty. Because kCELL operates in an online learning regime, these introspection quantities evolve through constant interaction with the environment and can be evaluated at every MPC iteration without retraining, posterior computations, or auxiliary optimization procedures.

Exploration can therefore be formulated as a deterministic, introspection-based regularization of the control objective, leading to the following augmented cost function

$$J\left(s_t, u_t\right) = \alpha J_{\text{task}}\left(s_t, u_t\right) + \lambda J_{\text{explore}}\left(s_t, u_t\right)$$

where J_{task} is the task-related cost function, J_{explore} the exploration penalty, and $\alpha \in \mathbb{R}$, $\lambda \in \mathbb{R}$ the adjustment weights to trade-off task exploitation and exploration.

4.2.1 Distance-based Competence

Unlike opaque black-box models, kCELL provides an explicit measure of its degree of knowledge about specific regions of the feature space through the spatial distribution of agents. To encourage exploration, the model must acquire data that lie outside the currently covered regions, i.e outside regions that have been sufficiently visited. The degree of knowledge associated with a state-action pair can be quantified by evaluating the Mahalanobis distance between the pair and the closest agents.

The objective is to deliberately drive the state-action trajectories away from the currently covered regions of the feature space to collect informative samples in poorly modeled areas. To achieve this, we introduce a distance-based penalty term $p_{\text{dist}}\left(x_t\right)$ that augments the MPC cost function by promoting exploration of unknown regions such as

$$p_{\text{dist}}\left(x_t\right) = -\sum_{i \in D_{\text{closest}}} w_i\left(x_t\right)\left(\left(x_t - \mu_i\right)^\top \Sigma_i^{-1}\left(x_t - \mu_i\right)\right) \tag{4.6}$$

$$= -\sum_{i \in D_{\text{closest}}} w_i\left(x_t\right) d\left(\mathcal{A}_i, x_t\right)^2 \tag{4.7}$$

where $x_t = [s_t, u_t]$, D_{closest} is the set of the k-closest agents, μ_i and Σ_i are the parameters of their activation function, $w_i\left(x_t\right) = \frac{\phi_i(x_t)}{\sum_j \phi_j(x_t)}$ are the weights depending on the activation of each agent on x_t.

Because the penalty is negative, minimizing the augmented MPC objective encourages trajectories that maximize the distance to the currently well-covered regions. in this context, $p_{\text{dist}} \ll 0$ means that x_t is currently not supported by the agents' training data. Conversely, $p_{\text{dist}} \approx 0$ indicates that x_t is close to a well-covered (known) region of feature space and is therefore likely to be already included in the current knowledge base.

4.2.2 Local Disagreement

In regions where multiple agents are close in feature space, the variance between their individual predictions serves as a proxy for local predictive inconsistency. High disagreement suggests that the local dynamics are either stochastic, noisy or that the model has not yet converged to a stable local representation.

The objective is to drive the state-action trajectories towards regions exhibiting high predictive disagreement to force the model to discover new areas or to converge in already known region of feature space. This mechanism is functionally analogous to uncertainty-driven exploration strategies proposed in [67, 61, 52, 14] while remaining fully intrinsic to the kCELL architecture.

To achieve this, we introduce a disagreement-based penalty term $p_{\text{disagree}}(x_t)$ that augments the MPC cost function by promoting exploration of highly uncertain regions of feature space, such as

$$p_{\text{disagree}}(x_t) = -\sum_{i \in D_{\text{closest}}} w_i(x_t) \left\| f(x_t) - f_i(x_t) \right\|^2 \tag{4.8}$$

where $x_t = [s_t, u_t]$, $D_{\text{closest}}$is the set of the k-closest agents, $f(x_t)$ is the global prediction of the system obtained by aggregating the predictions of the closest agents, $f_i(x_t)$ is the prediction of the i-th agent, $w_i(x_t) = \frac{\phi_i(x_t)}{\sum_j \phi_j(x_t)}$ are the weights depending on the activation of each agent on x_t.

4.2.3 Comparative Study

This section evaluates the effectiveness of the proposed introspection-driven exploration strategies based on distance-based competence and local model disagreement. The objective is to assess whether embedding these signals into the MPC objective leads to improved coverage of feature space in an online learning setup, compared to random uninform or purely exploitative exploration strategies.

Experimental Protocol

This experiment is conducted on the classical Pendulum control task available in gymnasium reinforcement learning library [71], which exhibits nonlinear dynamics and presents a nontrivial exploration challenge.

States are composed of the x, y 2D-coordinates of the tip of the pendulum and of $\dot{\theta}$ the angular velocity. For convenience reasons, in subsequent figures, states are also represented by θ, the angle and $\dot{\theta}$ to allow for 2D representations.

The pendulum is initialized at the bottom equilibrium with zero angular velocity ($\left\{\theta = \pi,\ \dot{\theta} = 0\right\}$) at the beginning of each episode. No initial state randomization is applied.

This way, high energy states are harder to reach, making the exploration harder for uninformed random strategies. The cost function to minimize to solve the task is the following

$$\min_{u_{0:T}} \sum_{t=0}^{T} \left((s_t - s_{\text{ref}})^\top Q (s_t - s_{\text{ref}}) + u_t^\top R u_t \right) + \left((s_{T+1} - s_{\text{ref}})^\top Q (s_{T+1} - s_{\text{ref}}) \right) \tag{4.9}$$

with $Q \in \mathbb{R}^{3\times3}$, $R \in \mathbb{R}^{1\times1}$, and s_{ref} the target state to reach for the pendulum which usually is the top neutral position.

The agent interacts with the environment for a total of 5000 time steps, divided into episodes of 500 steps. During interaction, a kCELL model is learned online from streaming data to predict the next state.

Trajectory optimization is performed using the Cross-Entropy Method (CEM) because this approach is stochastic. Using a population-based optimizer naturally introduces variability in candidate action sequecnes. This stochasticity is important in the early learning phase of kCELL, as it avoids situations where newly created agents would be initialized with near-zero action variance, which can greatly hurt local model learning.

Four exploration strategies are evaluated:

1. Distance intrinsic exploration, where the cost function to minimize is the distance-based competence penalty introduced in Section 4.2.1.

2. Disagreement intrinsic exploration, where the cost function to minimize is the disagreement-based penalty introduced in Section 4.2.2.

3. Pure exploitation, where only the original task cost is minimized without any exploration regularization.

4. Uniform random exploration, where actions are sampled uniformly at random within admissible bounds.

For all exploration strategies, the same kCELL architecture, with same hyperparameters and interaction budget is learned. The metrics measured in subsequent sections are averaged over 5 seeds to eliminate part of the stochastic bias induced by using stochastic optimizer and random exploration strategies. All strategies share identical model architecture, hyperparameters and interaction budget ensuring that observed differences arise solely from the exploration strategy.

Quantitative State-Space Coverage

To quantify the state exploration performance, the continuous state space is discretized into a regular grid of bins (50 bins per dimension). Each visited state is mapped to its corresponding bin, which is then marked as visited. State-space coverage is then defined as the ratio of visited bins to the total number of bins. The final coverage ratios are (averaged over 5 seeds), are reported in Table 4.1.

	Disagreement (ours)	Distance (ours)	Random Uniform	Full Exploitation
Coverage (in %)	$\mathbf{58.6 \pm 5}$	$\mathbf{57.9 \pm 3}$	49.2 ± 6	8 ± 4

Table 4.1: Measured state space coverage ratio (in %) after 5000 exploration steps. Best scores are highlighted in bold.

The results show that both introspection-driven strategies (distance-based competence and local disagreement) achieve significantly higher state-space coverage than the uninformed random uniform strategy under the same interaction budget: $\approx 10\%$ higher ratio for introspection-based strategies. Conversely, with pure exploitation strategy, where the controller optimizes only the task cost without any exploration regularization, results in the lowest coverage. This indicates that in the absence of prior knowledge of the dynamics, exploitation (i.e optimizing the task objective) alone is insufficient to drive the system toward diverse and informative regions of the state space.

Furthermore, The distance-based and disagreement-based strategies achieve comparable final coverage levels, indicating that both introspection signals are effective at promoting exploration beyond the regions initially supported by the data.

Evolution of Coverage during Exploration

Figure 4.6 shows the evolution of state-space coverage over time averaged over 5 seeds. The two introspection-driven strategies consistently dominate the random uniform baseline. Pure exploitation seems to remain confined to a narrow subset of states and fails to significantly increase coverage.

Both introspection-driven strategies show comparable growth in coverage over time. Minor differences in variability across seeds are observed but the mean coverage remains similar. This suggests that both strategies effectively promote exploration. Any apparent differences should be interpreted cautiously as they may not be statistically significant.

Overall, these results confirm that embedding introspection signals directly into the MPC objective yields an exploration advantage over uninformed stochastic action sampling.

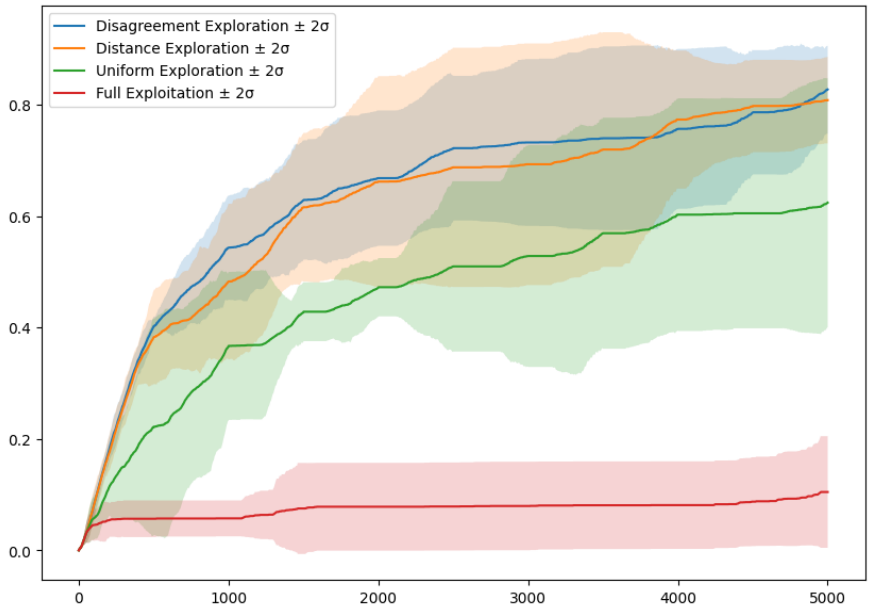


Figure 4.6: Evolution of the measured state space coverage ratio during exploration phase. x-axis corresponds to the number of steps and y-axis corresponds to the measured state space coverage ratio.

Spatial Distribution of Visited States

Figure 4.7 presents heatmaps of visited states in the $(\theta, \dot{\theta})$ space after the exploration phase. For all strategies, states corresponding to low angular velocities and angles that are close to the starting configuration ($\theta \approx \pi$) are visited more frequently, reflecting the energy barrier associated with reaching high energy states associated with throwing the pendulum tip all the way to the top.

Clear differences emerge between strategies. Both distance-based and disagreement-based explorations achieve substantially broader coverage, with frequent visits to high-energy states corresponding to near-upright pendulum configurations ($\theta \approx 0$ or 2π) and large angular velocities. In contrast, the random uniform strategy exhibits sparse and uneven coverage, with large regions remaining unvisited (especially in high energy regions). Pure exploitation remains almost entirely confined near the starting state. It consistently fails to overcome the energy barrier required to explore the upper half of the state space.

These observations highlight the limitations of uninformed random exploration in structured dynamical systems and demonstrate that introspection driven objectives allow the controller to deliberately seek out to reach informative regions of that are otherwise difficult to reach.

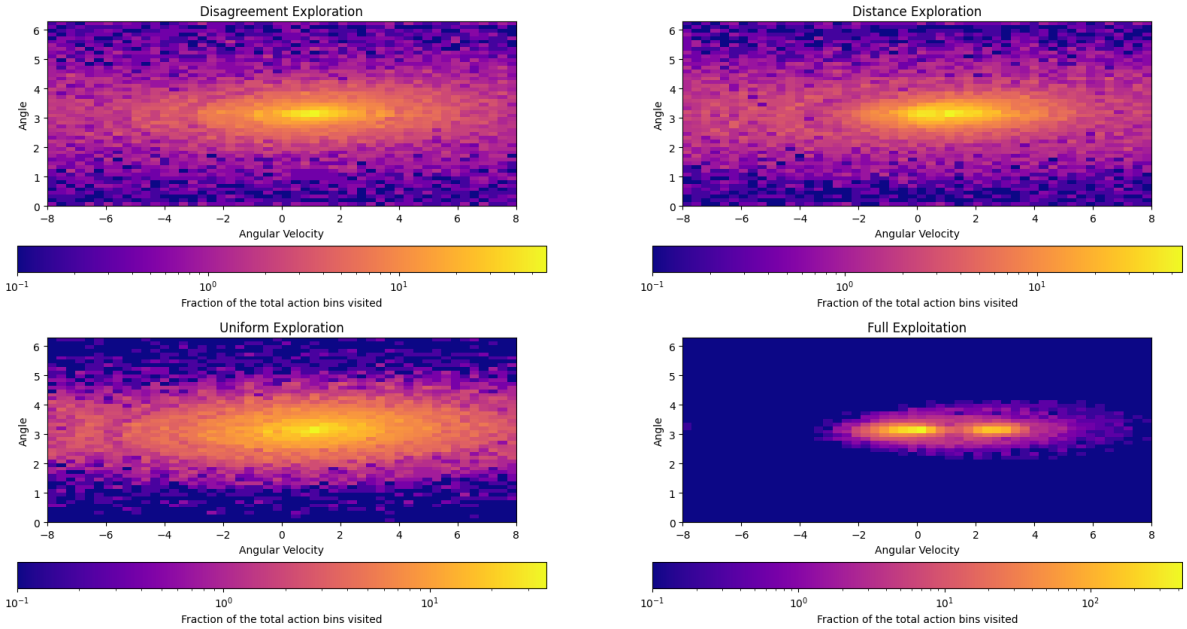


Figure 4.7: Obtained space coverage for the exploration strategies considered. *Top-left* corresponds to Disagreement strategy; *Top-right* corresponds to Distance strategy, *Bottom-left* corresponds to Random Uniform strategy; and *Bottom-right* corresponds to Full Exploitation strategy. The more yellow, the more the cell has been visited across the different seeds, conversely for blue colored cells. y-axis corresponds to the angle and x-axis corresponds to angular velocity.

Agent Activation Patterns and Local Geometry

Figure 4.8 illustrates the spatial organization of kCELL agents after the exploration phase. Figure 4.8a shows agent activations projected onto the Cartesian coordinates of the pendulum tip. Figure 4.8b displays activations in the $(\theta, \dot{\theta})$ space.

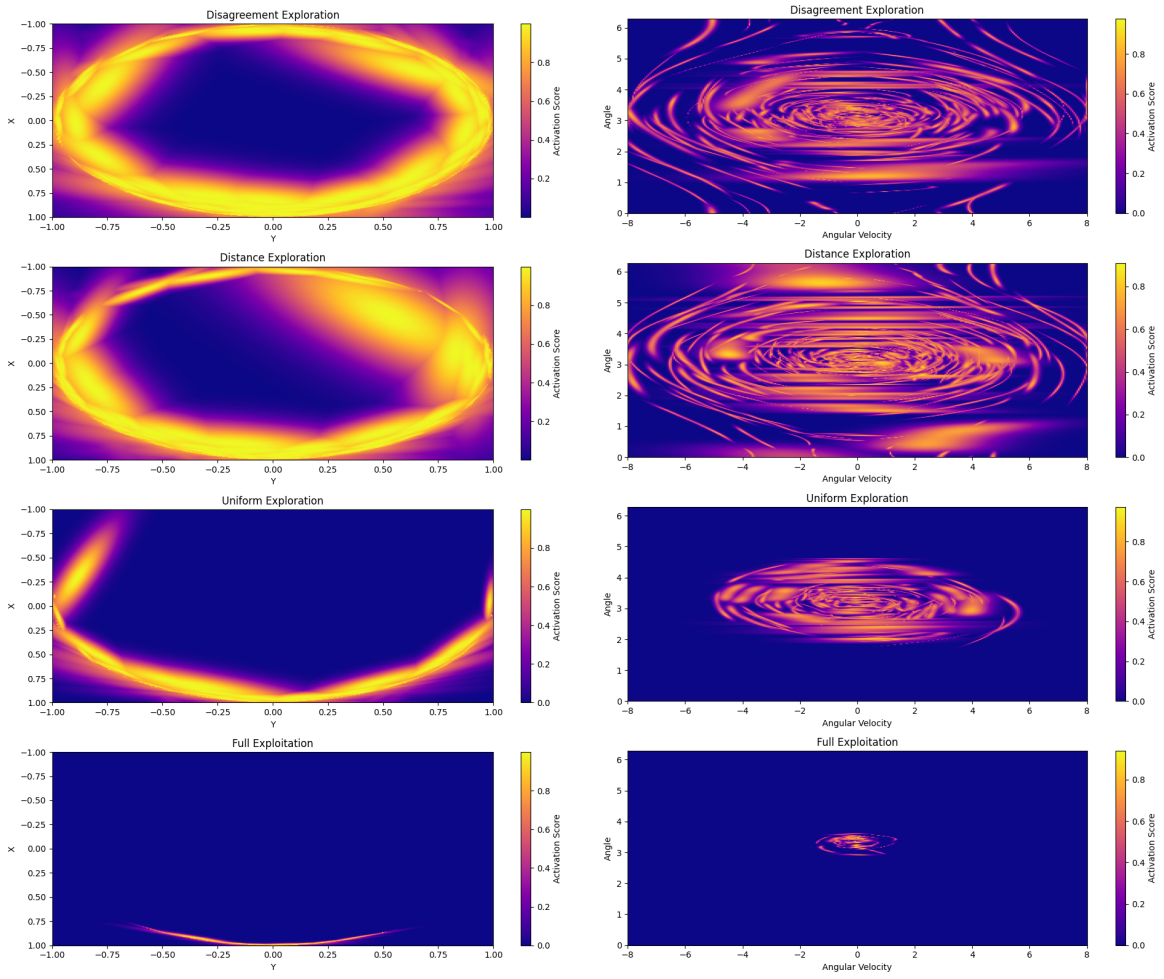
For the distance-based and disagreement-based strategises, agent activations cover the entire circular manifold corresponding to the pendulum's reachable positions, indicating that even high-energy and rarely visited states are locally modeled by dedicated agents. In contrast, with the random uniform and pure exploitation strategies, agent activations are concentrated in the lower-energy regions of the state space.

A broader diversity in agent sizes can be observed for the introspection-driven strategies. This is most likely due to kCELL agent creation mechanism, which creates agents from a minimum number of samples to ensure thet local models are quickly reliable. In high-velocity regimes or during abrupt torque changes such as strong braking events, state transitions are more widely distributed, leading to larger initial covariance estimates. These larger agents are not pathological artifacts but instead reflect an specific experience of data, here corresponding to regions of high local variability.

For example, in the $(\theta, \dot{\theta})$ projection, some agents appear elongated along one dimension especially for the Distance-based strategy. Analysis of their covariance matrices reveals that these agents correspond to sharp directional changes in the dynamics, typically induced by large control intpus applied against the direction of motion. As such, agent geometry seems to

provide an interpretable signature of local dynamical regimes encountered during exploration. The Distance-based strategy seeming to have more of these king of agents could be due to a more agressive exploration strategy that prioritizes more extreme action variations.

These activation patterns demonstrate that kCELL does not cover the state space uniformly but organizes local models in a manner that reflects the underlying geometry and variability of the dynamics encountered during exploration.



(a) (b)

Figure 4.8: Visualization of the activation patterns of agents for all considered strategies. Starting from the top, 1^{st} plot corresponds to the Disagreement strategy; 2^{nd} plot corresponds to the Distance strategy, 3^{rd} plot corresponds to the Random Uniform Strategy and finally 4^{th} plot corresponds to the Full Exploitation strategy. 4.8a shows the activation patterns of agents projected in the space of 2D coordinates of the tip of the pendulum ($x = \cos(\theta)$ and $y = \sin(\theta)$). 4.8b shows the activation patterns of agents projected in the space of Angle (y-axis) and Angular Velocity (x-axis).

Agent Population Dynamics

Figure 4.9 shows the evolutions of the number of agents during exploration. Both introspection-driven strategies lead to a substantially larger number of agents than the random uniform and pure exploration baselines. This increase is positively correlated with the observed improvement in state-space coverage.

While both introspection-driven strategies generate more agents overall, the disagreement-based strategy exhibits higher variability in agent population across the 5 considered seeds, which is consistent with the higher sensitivity of disagreement to local modeling inconsistencies. The more the space is covered, the more the local models can conflict locally and drive exploration to them for consolidation.

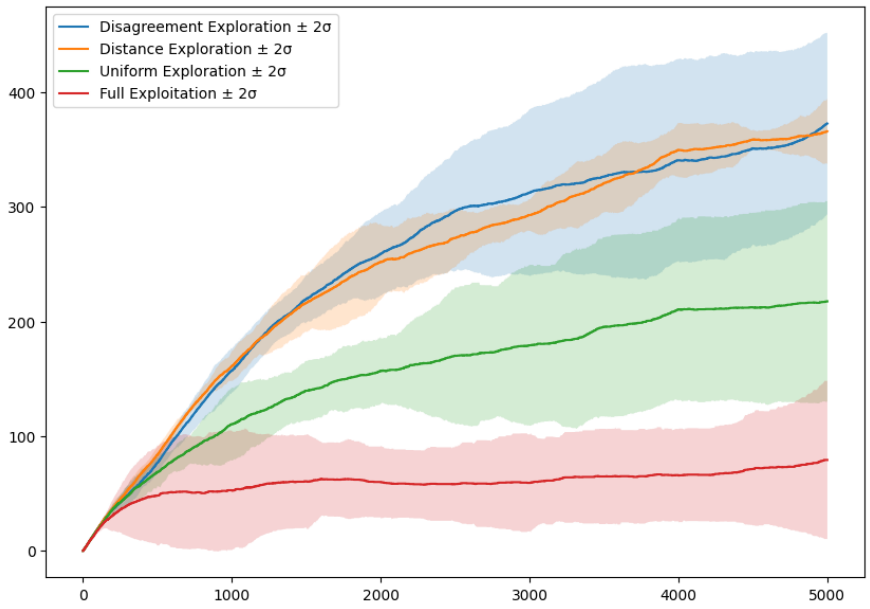


Figure 4.9: Evolution of agent population for the four considered strategies average over 5 seeds. y-axis corresponds to the number of agents and x-axis corresponds to the number of exploration steps.

Prediction Accuracy and Sample Efficiency

To assess whether introspection-driven exploration strategies translate into better models, we evaluate the predictive accuracy of kCELL after the exploration phase. For each exploration strategy, the learned model is evaluated on an exhaustive test set crafted specifically to cover a large portion of the state-action space. Model performance is quantified using the root mean squared error (RMSE) between predicted and ground-truth next states. Figure 4.10, presents a bar plot that reports the average RMSE across five seeds for the four exploration strategies.

Both introspection-driven strategies achieve substantially lower prediction error than the random uniform and pure exploitation baselines. In contrast, no statistically significant difference in RMSE is observed between the distance-based and disagreement-based strategies.

These results indicate that introspection-driven exploration improves the sample efficiency of online model learning in kCELL. While the two introspection signals lead to slightly different exploration dynamics, they results in comparable models in terms of predictive quality under the same interaction budget. This suggests that the shallow performance differences observed in state-space coverage are not likely associated with systematic differences in model accuracy under the considered interaction budget and evaluation protocol.

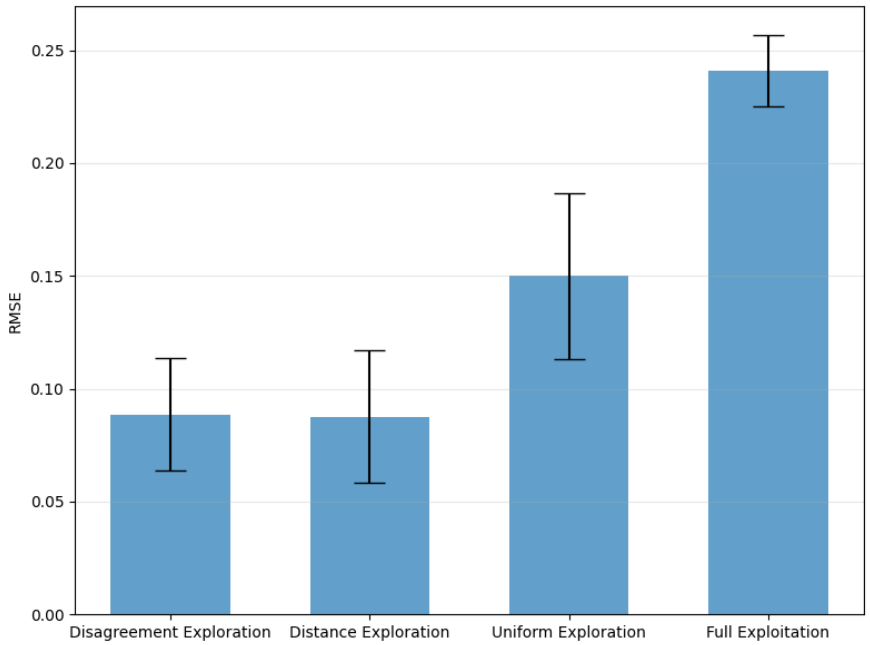


Figure 4.10: RMSE averaged over 5 seeds. y-axis is the RMSE and x-axis ticks corresponds to the corresponding exploration strategy.

Global Action Distribution

Although distance-based and disagreement-based strategies yield comparable coverage and predictive accuracy, minor differences in exploration dynamics were observed in coverage variability and in agent population statistics across seeds. Those variations are subtle and may reflect stochastic effects of the optimizer rather than systematic differences. To investigate the general behavior of the controller under introspection-driven exploration objectives, we analyzed the distribution of actions taken during exploration. Figure 4.11 presents kernel density estimates of the applied control inputs.

The resulting action distributions are very similar for both introspection-driven strategies. Three dominant modes can be observed centered around saturated control actions (-2, 2) and near-zero actions. This multimodal structure contrasts with the random uniform baseline. It indicates that both introspection-driven strategies induce structured non-uniform action profiles. The presence of repeated extreme actions suggests that introspection-driven objectives actively promote saturated actions in order to reach higher energy states rather than relying on diffuse stochastic perturbations.

These results suggest that the observed similarities in coverage and model accuracy are

more likely attributable to how these signals interact with local geometry and agent creation rather than fundamentally different global action sampling behaviors.

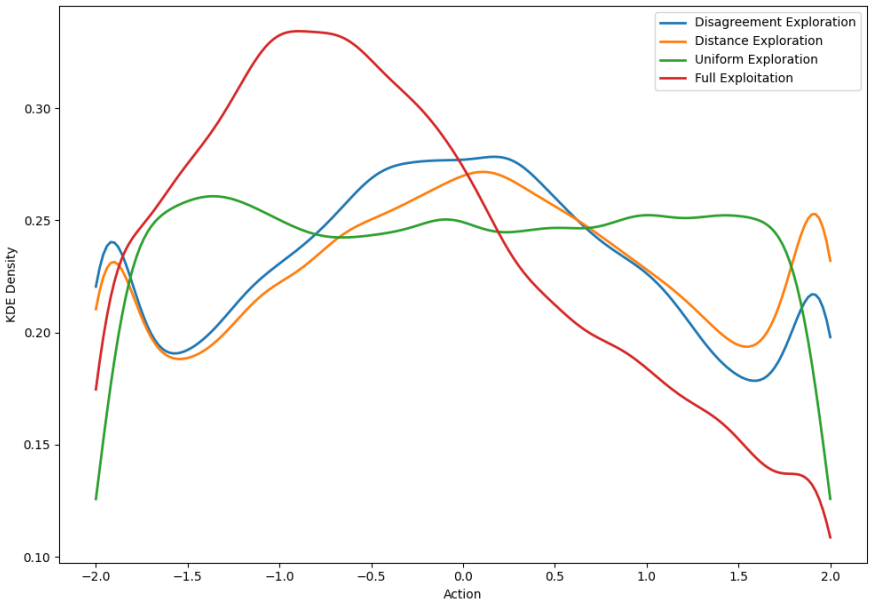


Figure 4.11: KDE density over actions taken during exploration for each considered exploration strategy. y-axis is the KDE density and x-axis ticks corresponds to the actions (in pendulum it corresponds to the applied torque).

Discussion

The experimental results presented in this section demonstrate that introspection-driven exploration strategies based on kCELL online modeling improve exploration efficiency in model-based control settings without prior knowledge of the system dynamics. By embedding distance-based competence and local model disagreement directly into the MPC objective, the controller is able to consistently escape well-modeled regions and acquire informative data in unknown regions of the state-action space.

Coverage metrics and spatial visualizations show that both introspection-driven strategies lead to a better coverage than uninformed random uniform exploration and pure exploitation. Introspection-driven strategies are able to overcome the energy barrier induced by the pendulum environment. High energy configurations are repeatedly visited despite being hard to reach for uninformed strategies. These results confirm that effective exploration cannot be achieved through randomly uniform actions alone, but instead requires additional motivations like model knowledge or uncertainty.

The analysis of global action distributions confirms that both introspection-driven strategies induce structured multimodal action profiles that facilitate exploration of high-energy states. The similarities observed in these distributions suggest that the two strategies behave comparably interms of global action patterns. Differences in coverage or agent population arise primarily from interactions with local agent geometry rather than fundamentally different action sampling.

At the agent level, activation patterns and population dynamics provide insights into the underlying mechanisms of introspection-driven exploration strategies. Increased agent creation correlates strongly with improved coverage. This indicates that exploration emerges from the interaction between the introspection objective and kCELL's agent creation process. Furthermore, agent geometry and size variability reflect local properties of the dynamics. For example, for high-velocity regimes, created agents tend to be larger or for sharp directional changes, agents displays a high variance on the action dimension. Each agent represents a singular experience of data. This highlights one of the key advantage of kCELL, which is its interpretability. Indeed, uncertainty and model structure are not hidden within opaque parameters but instead manifest explicitly through the spatial organization of agents.

Finally, the observed improvements in predictive accuracy confirm that introspection-driven exploration translates into concrete benefits for online model learning. Under a fixed interaction budget, broader and more structured exploration leads to more accurate learning dynamics. It reinforces the link between exploration strategy, model quality and downstream control performance.

Overall, kCELL's architecture allows to extract introspection signals that can be leveraged successfully to build efficient uncertainty-aware exploration mechanisms without prior knowledge about the dynamics of the environment. Those mechanisms do not rely on external stochastic processes or auxiliary uncertainty models.

Limitations

While the results demonstrate the effectiveness of introspection-driven exploration with kCELL, several limitations still stand and need to be acknowledged.

kCELL was introduced to address the overoptimistic spatial representations of oCELL (cf. Section 3.2.5) that originated from the use of orthotope based spatialization. While allowing more degrees of freedom and a better representative power, using Gaussian-based spatialization introduces its own set of constraints. Agents are spatialized using RBF kernels parameterized by the mean and covariance estimated from local ingested data. As a result, agent creation requires a minimum diversity level to avoid ill-conditioned covariance estimates. In practice, this makes early exploration phases or low-variability action regimes potentially problematic. On the other hand, while population-based optimizers such as CEM introduce stochasticity that partially mitigates this issue by inducing action variability, excessive dispersion of actions can lead to overly extended agents and interpolation errors. This effect can be observed in Figure 4.8b through some agents that appear elongated (especially for distance-based strategy) due to too spread out actions. These agents still represent valid local experiences but such behavior may pose challenges when extending the approach to more complex control environments.

The Gaussian assumption underlying agent spatialization restricts the ability of kCELL to faithfully represent complex or multimodal feature manifolds that cannot be represented by a single Gaussian kernel. The agent population can approximate such structures through multiple overlapping agents, but this approximation remains indirect. Extending the spatialization mechanism beyond Gaussian assumptions, for example by using kernel density estimates or mixture-based representations, could allow kCELL to better capture complex feature geometries.

The experimental evaluation is restricted to a low-dimensional control problem (3 state dimensions and 1 action dimension). The pendulum environment captures nontrivial exploration challenges such as energy barriers. However, scalability to higher-dimensional systems remains an open question for kCELL. In such settings, agent management, coverage estimation and computational overhead may require additional mechanisms such as dimensionality reduction (which we explore in Chapter 5).

Finally, this study considers distance-based competence and local disagreement as independent exploration objectives. A combined approach was not explored, primarily because thses introspection signals operate on very different scales and have distinct geometrical interpretations. Investigating principled combinations or adaptive weighting schemes between distance-based and disagreement-based objectives could reveal complementary effects and lead to more exhaustive exploration strategies.

Beyond exploration, introspection signals extracted from kCELL naturally lead to the design of conservative control strategies under learned dynamics. While exploring exploits these signals to deliberately seek poorly modeled regions, the same quantities can be repurposed during exploitation to bias the controller toward states where the model is locally reliable. In this setting, introspection costs no longer acts as penalties but as a confidence-aware bonuses that encourage trajectoires to remain close to existing agents, i.e close to the current knwoledge of the model.

4.3 Conservative MPC with kCELL

Exploration strategies relying on kCELL intrinsic multiagent structure have shown successful to drive exploration in an unknown environment, leading to better predictive learned dynamic models than when exploring with uninformed random strategies.

To achieve safe control with learned models, i.e reliability of controllers, the problem of compounding errors and uncertainty in the model predictions need to be addressed at exploitation time [13]. With inexact dynamic models, small errors on the first horizon steps leads to massive errors over time. Compounding errors are deeply linked to epistemic uncertainty (lack of knowledge about the underlying true dynamics). In [13], the authors discuss using Gaussian Processes (GPs) [75] or Bayesian Neural Networks [8] to quantify the prediction uncertainty and take it into account in optimization to obtain safe conservative control policies.

kCELL's spatialized agent-based architecture provides valuable introspection signals that can serve as proxies for epistemic uncertainty. Those introspection signals can be directly embedded into the MPC objective to regularize the behavior of the resulting policy by keeping the explored states at optimization time into the knowledge landscape of the model, avoiding wandering into unknown states that would induce large error compounding effects.

In this chapter we focus specifically on distance-based competence introspection signals that quantify how far a state lies from regions that are sufficiently covered by agents. It provides a geometrically grounded measure of epistemic uncertainty.

To bias the policy toward a more conservative knowledge-aware regime, an introspection-based regularization of the control objective is defined, leading to the following augmented

cost function

$$J(s_t, u_t) = \alpha J_{\text{task}}(s_t, u_t) + \lambda J_{\text{conservative}}(s_t, u_t)$$

where J_{task} is the task-related cost function, $J_{\text{conservative}}$ the conservative regularization term, and $\alpha \in \mathbb{R}$, $\lambda \in \mathbb{R}$ the adjustment weights to trade-off task exploitation and conservatism. In this context we define a conservative policy, as a policy that is biased toward minimizing epistemic uncertainty.

4.3.1 Distance-based Conservatism

Unlike opaque black-box models, kCELL provides an explicit measure of its degree of knowledge about specific regions of the feature space through the spatial distribution of agents. To encourage conservatism, the controller must keep evaluated solutions and state trajectories close to the covered regions of the feature space. In kCELL, the degree of knowledge associated with an input x_t can be quantified by evaluating the Mahalanobis distance between x_t and the closest agents.

In kCELL one of the structural hypothesis is that agents are local experts on the function to approximate and that the closer an input is to the center of an agent, the most probable the prediction will be accurate. Thus, the objective is to deliberately drive the trajectories close to the currently covered regions, that is to say, closer to the center of agents.

To achieve this, we introduce a distance-based regularization term $q_{\text{dist}}(x_t)$ that augments the MPC cost function by promoting conservative behaviors such as

$$q_{\text{dist}}(x_t) = \sum_{i \in D_{\text{closest}}} w_i(x_t) \left((x_t - \mu_i)^\top \Sigma_i^{-1} (x_t - \mu_i) \right) \tag{4.10}$$

$$= \sum_{i \in D_{\text{closest}}} w_i(x_t)\, d(\mathcal{A}_i, x_t)^2 \tag{4.11}$$

where $x_t = [s_t, u_t]$, D_{closest} is the set of the k-closest agents, μ_i and Σ_i are the parameters of their activation function, $w_i(x_t) = \frac{\phi_i(x_t)}{\sum_j \phi_j(x_t)}$ are the weights depending on the activation of each agent on x_t.

Because the penalty is positive, minimizing the augmented MPC objective encourages trajectories that minimize the distance to the currently well-covered regions (in constrast to what is done for exploration in Section 4.2.1). in this context, $q_{\text{dist}} \gg 0$ means that x_t is currently not supported by the agents' training data. Conversely, $q_{\text{dist}} \approx 0$ indicates that x_t is close to a well-covered (known) region of feature space and is therefore likely to have been encountered during training. As q_{dist} is convex, it can easily be embedded for optimization with fast conic solvers like OSQP [68] within the context of Sequential Quadratic Programming (SQP) optimization framework.

4.3.2 Comparative Study

This section evaluates the effectiveness of the proposed introspection-driven regularization term based on distance to agents in a learned kCELL system through a low-dimensional experiment. The objective is to assess whether embedding this regularization term into the

MPC objective yields better known trajectories, less compounding errors and if there is a tradeoff between conservativeness and task resolution performance.

Experimental Protocol

This experiment evaluates the impact of introspection-driven conservative regularization on the reliability of trajectory optimization using a kCELL learned dynamics model within a MPC scheme. The objective is to assess whether considering the spatialization of agents of a kCELL model during optimization reduces extrapolation errors and improves robustness of long-horizon predictions.

The experiment is conducted on the classical Pendulum control task available in gymnasium reinforcement learning library [71] which exhibits nonlinear dynamics. States are composed of the x, y 2D-coordinates of the tip of the pendulum and of $\dot{\theta}$ the angular velocity.

A kCELL model is learned online from streaming interaction data. The population of linear agent predictors is spatialized in the state space (not including actions). This design choice is motivated by empirical observations on the pendulum task. Indeed, spatializing agents jointly over state and action dimensions leads to slower stabilization of agent population, whereas state-only spatialization resulted in faster convergence of local models and more stable agent coverage. Given the low-dimensional action space and the smooth dynamics, state-based spatialization provides a favorable tradeoff between model predictive accuracy and sample efficiency.

To ensure a sufficient coverage of the state space, an informed finetuned stochastic exploration policy based on Ornstein-Uhlenbeck (OU) noise is used to interact with the simulation environment to collect learning data. This exploration phase to build a kCELL model is conducted for a fixed interaction budget of 30k steps. The learned dynamics model is frozen and used subsequently for evaluation. No further learning occurs during evaluation, ensuring that observed effects are attributable to the optimization strategy rather than continued model adaptation.

Control actions are obtained using a MPC scheme. Action sequences are optimized over a finite horizon using the learned kCELL dynamics. Trajectory optimization is performed using SQP with OSQP conic solver to solve the local Quadratic Programs at each iteration. Using SQP, a gradient-based optimizer, allows to isolate the effect of the proposed regularization without introducing stochastic variability which is inherent to population-based optimization methods.

The baseline optimization objective, i.e the cost function to minimize to solve the task is the following

$$\min_{u_{0:T}} \sum_{t=0}^{T} \Big((s_t - s_{\text{ref}})^\top Q\,(s_t - s_{\text{ref}}) + u_t^\top R u_t \Big) + \Big((s_{T+1} - s_{\text{ref}})^\top Q\,(s_{T+1} - s_{\text{ref}}) \Big) \tag{4.12}$$

with $Q \in \mathbb{R}^{3\times3}$, $R \in \mathbb{R}^{1\times1}$, and s_{ref} the target state to reach for the pendulum which usually is the top neutral position.

To this objective, an additional conservative regularization term q_{dist} is introduced (cf Section 4.3.1). This regularization term is weighted by a coefficient $\lambda \geq 0$, penalizing predicted

trajectories that run accross low coverage regions of the state space, envouraging solutions that remain within areas well covered by the agent. The goal is to stay close to what the model knows.

The evaluation is conducted for multiple values of λ. We refer to the unregularized case $\lambda = 0$ as the baseline. To evaluate robustness and reliability independently of exploration stochasticity, control performance is assessed from an exhaustive set of initial states sampled over a grid in the state space. For each initial state, the predictions produced by kCELL with the found solution are compared against the ground-truth trajectory. This allows direct measurement of prediction error against the true dynamics, also allowing to assess the compounding effects on a given horizon. All evaluations are performed using identical MPC horizons ($T = 20$), cost function, solver parameters and constraints across regularization settings.

Validation of Regularization Behavior

To validate the behavior of the regularization term, we measure the mean agent activation along optimized trajectories. Activation of agents serves as a proxy for epistemic uncertainty by measuring the distance to the closest experts in state space. Figure 4.12 presents boxplots of the mean agent activation for each considered values of λ.

As λ increases, we observe a clear increase in mean agent activation indicating that the optimizer progressively favors trajectories that remain closer to the centers of kCELL agents, i.e within regions that are well supported by training data. This behavior is consistent with the expected behavior of the regularization term q_{dist}. The optimization landscape is effectively reshaped.

In addition to that, increasing λ also yields a noticeable reduction in the spread of the activation distributions across initial conditions. This reduced variability indicates that under stronger regularization trajectories initialized in different regions are increasingly constrained to remain within similarly well-supported regions. This effect suggests that the regularization not only increases overall conservatism but also homogenizes the level of model support encountered along optimized trajectories.

These results validate the behavior of the introspection-driven regularization mechanism. The regularizer acts as intended by increasing the proximity of optimized trajectories to kCELL agent centers and reduces variability in model competence across initial conditions.

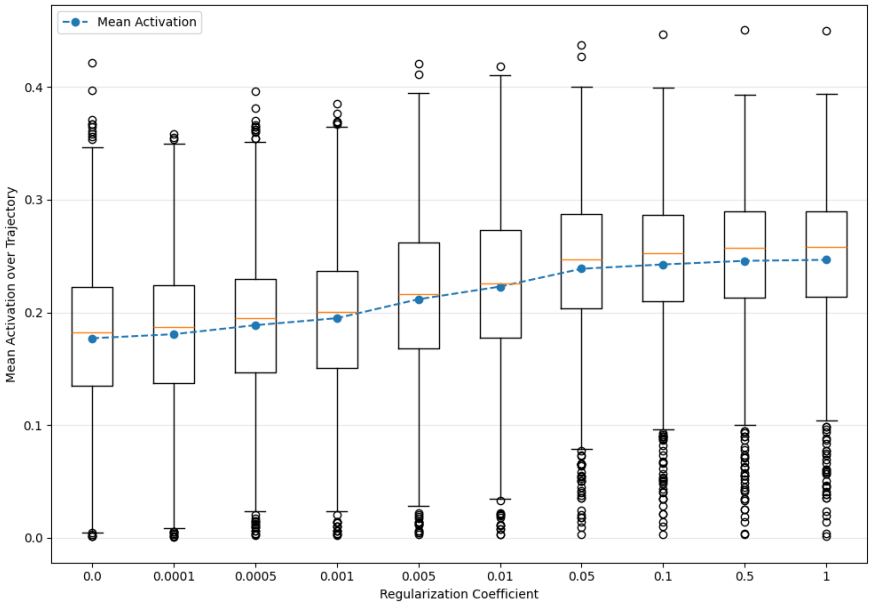


Figure 4.12: Boxplots of the mean activation over optimized trajectories for various values of regularization coefficient λ. y-axis corresponds to the mean activation over optimized trajectories and x-axis corresponsd to values of λ. The blue curve corresponds to the global mean and the orange bars correspond to the median value.

Impact on Prediction Accuracy

Figure 4.13 presents boxplots of the mean absolute error over trajectories for various values of regularization coefficient λ. For each value of regularization coefficient λ, along each optimized rollout, the predicted state transitions produced by kCELL are compared against ground-truth trajectories. Prediction error is computed and aggregated over the horizon.

As the regularization coefficient increases, a downward shift of the error distributions can be observed. Higher values of λ correspond to lower median and mean prediction errors indicating that trajectories optimized under strong regularization are predicted more accurately on average by the learned model. This behavior is consistent with the core hypothesis of kCELL that stipulates that regions of high agent activation correspond to areas where local linear models provide the most reliable approximations of the true dynamics.

This reduction in prediction error does not arise from any modification of the learned kCELL model itself which remains fixed during evaluation. Instead, it is a direct consequent of the optimizer preferentially selecting trajectories that remain within regions of the state space that are well supported by training data. Thus, the regularization acts as a mechanism for smarter model usage rather than model improvement. By penalizing trajectories that lie too much in lowly covered regions, the optimizer performs a form of constrained optimization where the constraint is the validity of the local linear approximation.

In addition to that, the downward shift in mean absolute error distributions for increasing values of λ is accompanied to a decrease in the spread of the error distributions. This indicates that large mean prediction errors become progressively less frequent as regularization strength increases. This suggests that distance-based regularization not only improves aver-

age prediction accuracy but also reduces the occurence of extreme prediction failures across considered initial conditions. This effect is particularly relevant in terms of reliability of the model as it seems to successfully prevent to some extent, the controller to exploit faulty out of distribution trajectories.

Overall, these results confirm that the spatial organization of agents in a kCELL model encodes meaningful information about local model reliability. This information can be effectively exploited during trajectory optimization. By encouraging trajectories to remain within regions of high agent coverage, the proposed regularization yields more accurate and more consistent predictions. This way it improves the reliability of model-based control with learned dynamics.

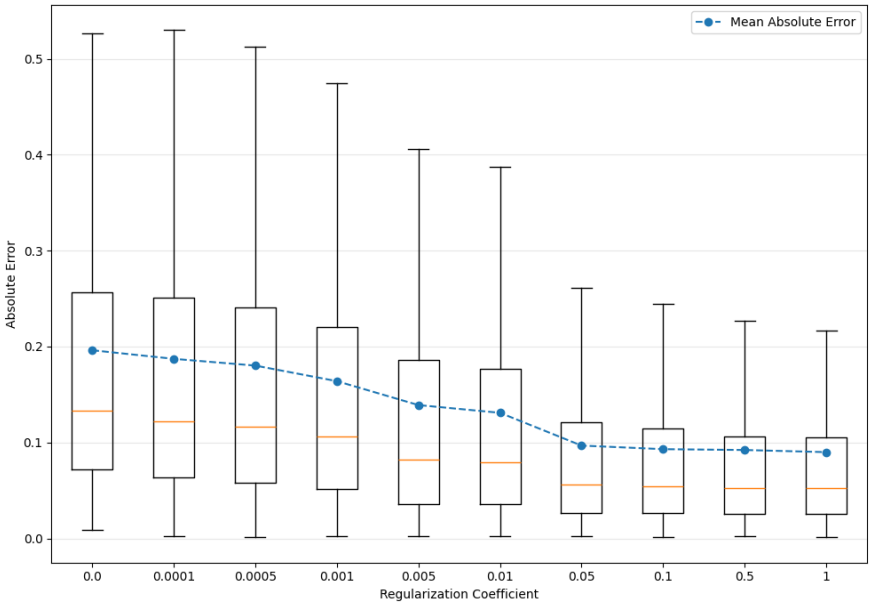


Figure 4.13: Boxplots of the mean absolute error over optimized trajectories for various values of regularization coefficient λ. y-axis corresponds to the mean absolute error over optimized trajectories and x-axis corresponds to values of λ. The blue curve corresponds to the global mean and the orange bars correspond to the median value.

Impact on Compounding Errors

One of the main limitations of learned dynamics models in model-based control is their susceptibility to error accumulation over long prediction horizons. Even small local inaccuracies can compound across successive rollout steps rapidly degrading the prediction quality and optimization results in the process. Figure 4.14 shows the evolution of the mean absolute prediction error over the prediction horizon for various values of λ.

In the absence of regularization, for the baseline, the prediction error increases rapidly with the horizon length reflecting the compounding effects of model errors in multi-step rollouts. This behavior highlights represents a challenge in standard MPC schemes based on learned dynamics models when operating with out of distribution data [13].

As λ values increase, the rate at which prediction error grows with the horizon steps is substantially reduced. Stronger regularization leads to consistently lower prediction errors

especially for more distant horizon steps. This confirms that the primary benefit of the proposed regularization lies in mitigating compounding errors rather than improving one-step prediction accuracy.

Overall, these results show that this introspection-based regularization term improves the reliability of long-horizon predictions in MPC. By maintaining the trajectory within highly covered regions, the error propagation is dampened, preventing the exponential divergence typically observed in the unregularized case over the horizon. It reduces the myopic behavior induced by error-prone rollouts and enhances the practical usability of kCELL learned dynamics models for more reliable control policies.

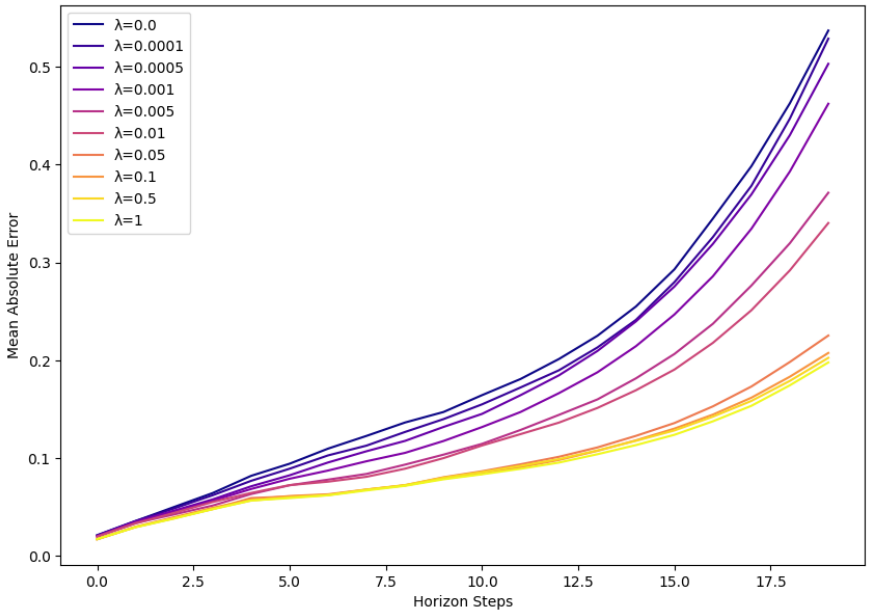


Figure 4.14: Evolution of the mean absolute error over the prediction horizon for various values of regularization coefficient λ. y-axis corresponds to the mean absolute error over optimized trajectories and x-axis corresponds to the horizon steps. Each colored curve corresponds to a given value of λ. The $\lambda = 0$ curve corresponds to the unregularized baseline case.

Cost-Accuracy Tradeoff

While the introspection-based regularization term improves the reliability of predictions in a model-based control setting, such conservatism is expected to impact global task solving performance.

Figure 4.15 show boxplots of the cumulative task cost over trajectories for various values of λ. As λ increases, the mean cumulative cost slightly increases compared to the unregularized baseline ($\lambda = 0$). This indicates that the controller increasingly sacrifices task optimality in order to remain within regions of the state space that are well supported by the learned kCELL model. In addition to that, the spread of the cost distributions decreases with stronger regularization. This suggests that conservative behavior leads to more uniform, less optimal, performances across initial conditions.

Figure 4.16 shows the evolution of the mean cumulative cost over the horizon steps for various values of λ. Higher regularization coefficiens λ resulted in higher accumulated costs compared to the baseline. This effect is more pronounced for longer horizons. It indicates that conservative trajectory selection restricts the optimizer's ability to exploit agressive control strategies that rely extensively on extrapolation in predictions.

To explicitely relate predictive performance to task performance, Figure 4.17 presents a scatter plot of mean absolute prediction error over cumulative cost for various values of λ. By interpolating a cubic polynomial on those points, a nonlinear relationship can be observed between the prediction error and cumulative costs. Indeed, trajectories associated with lower prediction errors tend to incur higher costs. The interpolated curve exhibits a steep initial decrease in cost as prediction error increases, followed by a more gradual regime.

Overall, for this specific control task, these results reveal an explicit reliability-performance tradeoff induced by the introspection-based regularization term we introduced. Stronger regularization improves prediction accuracy and reduces error accumulation at the expense of control optimality. The shape of the tradeoff shown in Figure 4.17 suggests that it is possible to find intermediate values of the regularization coefficient λ that yields substantial gains in predictive accuracy for a limited increase in cost. Further works on different control tasks should be carried on to find a practical tuning mechanism for the conservative coefficient.

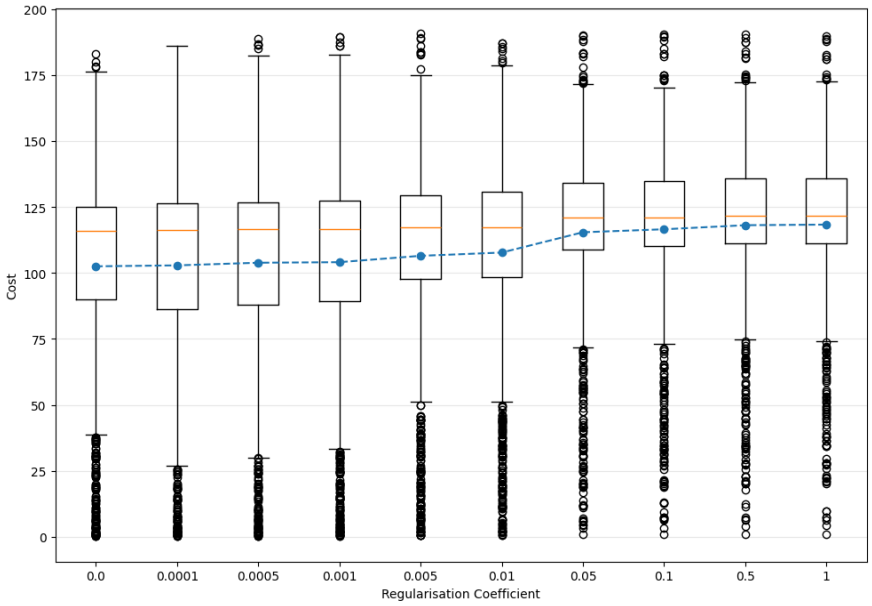


Figure 4.15: Boxplots of the cumulative costs over optimized trajectories for various values of regularization coefficient λ. y-axis corresponds to the cumulated cost over optimized trajectories and x-axis corresponds to values of λ. The blue curve corresponds to the global mean and the orange bars correspond to the median value.

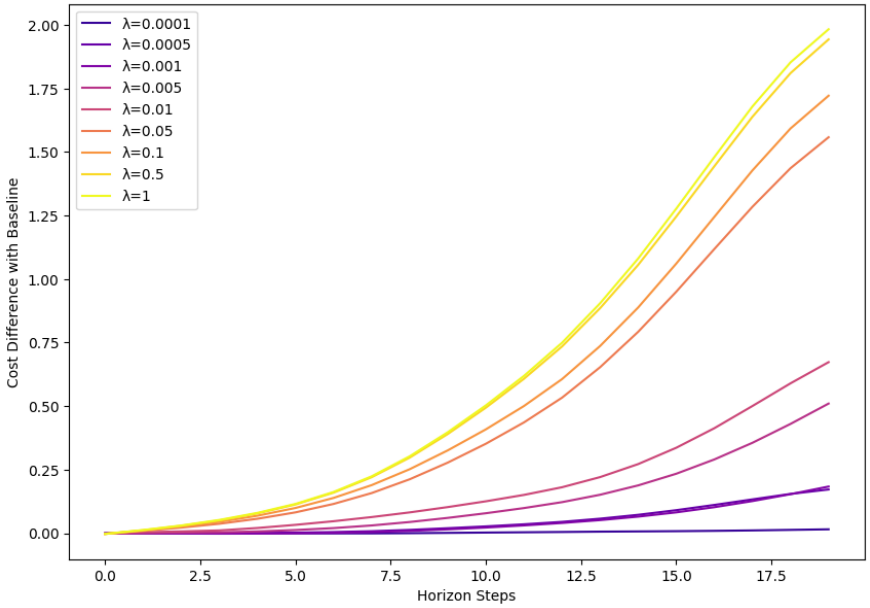


Figure 4.16: Evolution of the cumulative cost over the prediction horizon for various values of regularization coefficient λ. y-axis corresponds to the cumulative cost over optimized trajectories and x-axis corresponds to the horizon steps. Each colored curve corresponds to a given value of λ. The $\lambda = 0$ curve corresponds to the unregularized baseline case.

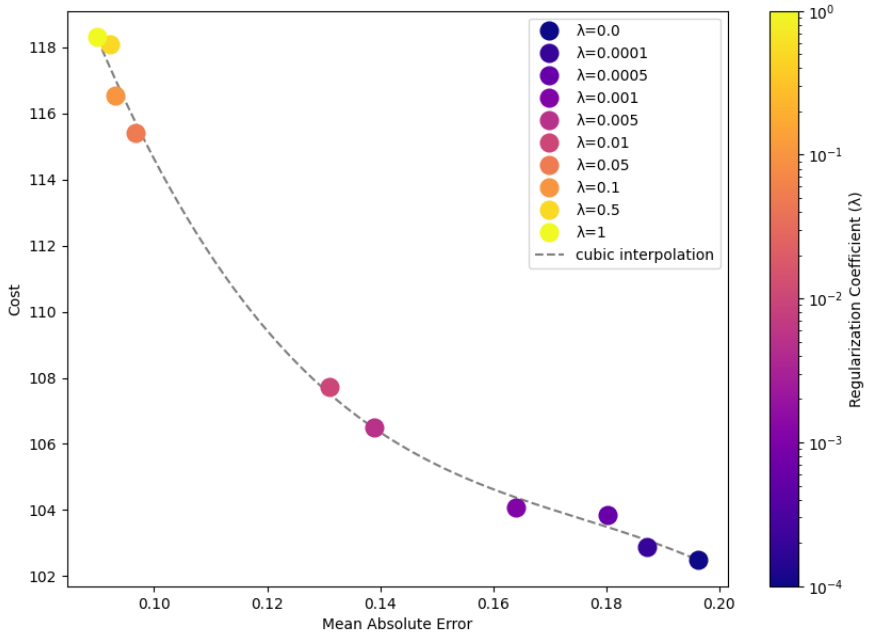


Figure 4.17: Visualization of cumulative cost against mean absolute error. y-axis corresponds to the cumulative costs and x-axis corresponds to the mean absolute errors over trajectories. Each colored point is associated to a value of λ. The greyed out line corresponds to a cubic interpolation.

Discussion

This experiment shows that the spatial organization of agents in kCELL can be directly exploited during trajectory optimization to improve the reliability of model-based control with a learned kCELL model. A distance-based regularization term derived from agent coverage is embedded into the MPC objective and it effectively bias the optimizer toward regions of the state space where the learned dynamics are supported by training data.

The results show that this regularization method leads to trajectories that remain closer to agent centers, reduced prediction errors and attenuated error compounding over horizon steps. These effects are observed without modifying the learned kCELL model. The improvement arises solely from how the model is exploited by the optimizer, rather than from improved accuracy that would be obtained by continuous learning. The proposed regularization mechanism does not attempt to make the model more accurate. Instead, it exploits epistemic awareness to make the better off of it.

Moreover, these results provide empirical support for the local expertise hypothesis that represents the core of kCELL. A correlation between activation of agents and reduced prediction error can be observed. It validates the assumption that local linear models are more and more reliable the closest the features are to their center. The regularization mechanism transforms this hypothesis into a control-relevant advanatage.

By increasing the regularization coefficient, the reduction in the growth rate of prediction error across the horizon steps improves largely the longer the horizon. In this case, it seems that the main effect of this introspection-based regularization approach lies in mitigatin erorr compounding effects. The regularization term prevents the optimizer from relying too much on aggressive extrapolated trajectories. This way, the controller looks to avoid MPC instabilities that arise from inaccurate predictions and misguided optimization. The pendulum environment is forgiving, because due to inertia, an unregularized MPC controller can recover from mistakes. However, in more complex control environments with less smooth dynamics this might not be the case and a regularization strategy could prove useful.

At the same time, theresults show a tradeoff between predictive accuracy and task performance in term of cumulated costs. Increasing conservatism leads to higher cumulative costs and lower predictive error. This behavior is expected and results directly from enforcing knowledge-aware regularization to the controller. The regularization acts as a soft feasibility constraint on the validity of the model, considering that unknown regions are somewhat unfeasible or dangerous. This is similar to trust-region methods in nonlinear optimization [21, 62].

Figure 4.17 showed that this tradeoff seems nonlinear. Reductions in prediction error can be obtained for relatively small increases in cost, suggesting that intermediate regularization strengths may offer a practical compromise between safety and performance. This observation is relevant for real-world applications where safety is required. In these cases, slight suboptimality is acceptable in exchange for improve reliability and reduced risk of catastrophic failure.

Overall, this study shows that kCELL's interpretability properties can be embedded into optimization pipelines. The proposed regularization mechanism consistures a principle way to translate structural transparency into conservative control behavior. This approach is a little step further towards reliable (i.e safe) model-based control based on learned dynamics

models.

Limitations

The comparative study that we conducted as well as the proposed introspection-based regularization mechanism exhibit some limitations that should be acknowledged.

In the presented experiments, proximity to agent centers correlates well with prediction accuracy. However, the distance-based competence measure used in this study is only a proxy for epistemic uncertainty. It is not a calibrated uncertainty estimate. As such, this regularization approach should be interpreted as a heuristic for risk reduction rather than a formal safety guarantee. Its effectiveness deeply depends on the structure and distribution of agents. No guarantees can be provided at the moment if the agent coverage is insufficient or poorly aligned with the true dynamics of the system.

The experiment has been conducted on a low-dimensional dynamical system (the pendulum) with smooth dynamics. As pointed out in Section 3.3.6, in low dimensional case, Mahalanobis distances (that are used for regularization) are not too distorted and remain meaningful. In higher dimensional cases the agent coverage might become more sparse. The impact of dimensionality on the effectiveness of the proposed regularization has not been studied in this chapter and remains an important limitation. This regularization approach might require additional mechanisms to remain effective.

We made the choice of spatializing agents only in the state space for convenience reasons to stabilize faster the agent population at learning time. This choice limits expressiveness of agents regarding actions. While this design choice is appropriate for the pendulum control task, it may not generalize to other dynamical systems. In such cases, regularization in joint state-action space could be required.

Another limitation relates to the density and sparsity of agent population. The conducted experiment rely on a single learned kCELL model trained with a fixed exploration budget. The effect of the sparsity of agent population on the conservative regularization behavior has not beed studied. Training multiple models with varying agent densities would be necessary to better characterize how coverage quality impacts the reliability gains.

In this work we show that it might be possible to find a value for the regularization coefficient λ that could represent a satisfying tradeoff between performance loss and increased predictive accuracy. However, we do not provide a principled way to tune the λ regularization coefficient. It must be done manually depending on the task nature and on the scale of task cost. Further investigations across different control environments are needed to extract practical insights and rules to chose the value of λ depending on the problem characteristics as well as performance and predictive accuracy requirements.

Finally, the proposed approach improves the reliability of the controller but does not ensure safety. The regularization reduces the likelihood of failure due to prediction errors by discouraging extrapolation but it does not prevent it entirely.

4.4 Towards Safe Explainable Control under 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 [75] 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) [38] 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 [5]. 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 $O(NM^2)$ [66, 49].

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 [69, 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] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[4] Julian Berberich and Frank Allgöwer. An overview of systems-theoretic guarantees in data-driven model predictive control. *Annual Review of Control, Robotics, and Autonomous Systems*, 8(1):77–100, 2025.

[5] 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.

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

[7] 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.

[8] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[9] 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.

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

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

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

[13] Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annu. Rev. Control Rob. Auton. Syst.*, (Volume 5, 2022):411–444, May 2022.

[14] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

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

[16] 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.

[17] Runqi Chai, Al Savvaris, Antonios Tsourdos, Senchun Chai, and Yuanqing Xia. A review of optimization techniques in spacecraft flight trajectory design. *Progress in aerospace sciences*, 109:100543, 2019.

[18] 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.

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

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

[21] Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust Region Methods*. SIAM, 2000.

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

[23] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

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

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

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

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

[28] 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.

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

[30] Sean Gillies. Rtree: Spatial indexing for Python GIS.

[31] 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.

[32] Luigi Grippo, Francesco Lampariello, and Stefano Lucidi. A nonmonotone line search technique for Newton's method. *SIAM journal on Numerical Analysis*, 23(4):707–716, 1986.

[33] Ralf Hartmut Güting. An introduction to spatial database systems. *the VLDB Journal*, 3(4):357–399, 1994.

[34] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

[35] 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.

[36] 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.

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

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

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

[40] 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.

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

[42] Nikolaos Kouiroukidis and Georgios Evangelidis. The effects of dimensionality curse in high dimensional knn search. In *2011 15th Panhellenic Conference on Informatics*, pages 41–45. IEEE, 2011.

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

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

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

[46] 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.

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

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

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

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

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[52] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017.

[53] 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.

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

[55] Krupa Prag, Matthew Woolway, and Turgay Celik. Toward data-driven optimal control: A systematic review of the landscape. *IEEE Access*, 10:32190–32212, 2022.

[56] 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.

[57] James Blake Rawlings, David Q Mayne, Moritz Diehl, et al. Model predictive control: Theory, computation, and design, vol. 2. *Madison, WI: Nob Hill Publishing*, 2017.

[58] 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.

[59] 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.

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

[61] Alberto Viseras Ruiz and Calin Olariu. A general algorithm for exploration with gaussian processes in complex, unknown environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3388–3393. IEEE, 2015.

[62] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[63] 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.

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

[65] 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.

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

[67] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[68] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

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

[70] 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.

[71] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[72] George E Uhlenbeck and Leonard S Ornstein. On the theory of the Brownian motion. *Physical review*, 36(5):823, 1930.

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

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

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

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

[77] 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.

[78] 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.