UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUCAS NUNES ALEGRE

Minimum-Delay Adaptation in Non-Stationary Reinforcement Learning via Online High-Confidence Change-Point Detection

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Bruno C. da Silva

Porto Alegre
November 2020

"Changes aren't permanent,

but change is."

— NEIL PEART

ACKNOWLEDGMENTS

First, I would like to thank the professors Bruno C. da Silva and Ana L. C. Bazzan. Their feedback, teaching and advice were not limited to my work as a scientific initiation researcher, but provided me models for the kind of scientist I want to become. Today, I am also grateful for being able to consider them my friends.

I would like to thank my family for the unconditional support during all my years of study. I have no doubt that all my accomplishments would have not been possible without the education, care and affection they have provided me.

The support of my friends was also fundamental to keep me motivated and able to deal with all challenges along the way. I am extremely lucky to count on the friendship of Guilherme Haetinger, Alexandre Ilha and Catarina Nogueira, all of who were there for me since the first day at UFRGS. In special, I have no words for the loving company of Letícia Haas during the last two years. Thank you for being on my side. I love you all.

ABSTRACT

Non-stationary environments are challenging for reinforcement learning algorithms. If the state transition and/or reward functions change based on latent factors, the agent is effectively tasked with optimizing a behavior that maximizes performance over a possibly infinite random sequence of Markov Decision Processes (MDPs), each of which drawn from some unknown distribution. We call each such MDP a *context*. Most related works make strong assumptions such as knowledge about the distribution over contexts, the existence of pre-training phases, or *a priori* knowledge about the number, sequence, or boundaries between contexts. We introduce an algorithm that efficiently learns policies in non-stationary environments. It analyzes a possibly infinite stream of data and computes, in real-time, high-confidence change-point detection statistics that reflect whether novel, specialized policies need to be created and deployed to tackle novel contexts, or whether previously-optimized ones might be reused. We show that *(i)* this algorithm minimizes the delay until unforeseen changes to a context are detected, thereby allowing for rapid responses; and *(ii)* it bounds the rate of false alarm, which is important in order to minimize regret. Our method constructs a mixture model composed of a (possibly infinite) ensemble of probabilistic dynamics predictors that model the different modes of the distribution over underlying latent MDPs. We evaluate our algorithm on high-dimensional continuous reinforcement learning problems and show that it outperforms state-of-the-art (model-free and model-based) RL algorithms, as well as state-of-the-art meta-learning methods specially designed to deal with non-stationarity.

Keywords: Reinforcement learning. non-stationarity. model-based RL. change-point detection.

RESUMO

Ambientes não-estacionários são desafiadores para algoritmos de aprendizado por reforço. Se as funções de transição de estado e/ou recompensa mudam com base em fatores latentes, o agente é efetivamente encarregado de otimizar um comportamento que maximize o desempenho em uma sequência aleatória possivelmente infinita de Processos de Decisão de Markov (MDPs), cada um deles amostrado de uma distribuição desconhecida. Chamamos cada MDP de um *contexto*. A maioria dos trabalhos relacionados faz suposições fortes, como conhecimento sobre a distribuição sobre os contextos, a existência de fases de pré-treinamento ou conhecimento *a priori* sobre o número, a sequência ou os limites entre os contextos. Nós introduzimos um algoritmo que eficientemente aprende políticas em ambientes não-estacionários. Ele analisa um fluxo possivelmente infinito de dados e computa, em tempo real, estatísticas de alta confiança para detecção de pontos de mudança que refletem se novas políticas especializadas precisam ser criadas para lidar com novos contextos ou se políticas previamente otimizadas podem ser reutilizadas. Nós demonstramos que este algoritmo *(i)* minimiza o atraso até que mudanças imprevistas em um contexto sejam detectadas, permitindo assim respostas rápidas; e *(ii)* que limita a taxa de alarmes falsos, o que é importante para minimizar o *regret*. Nosso método constrói uma mistura de modelos composta por um conjunto (possivelmente infinito) de preditores probabilísticos da dinâmica que modelam os diferentes modos de distribuição sobre MDPs latentes subjacentes. Nós avaliamos nosso algoritmo em problemas de aprendizado por reforço contínuos e de alta dimensionalidade e mostramos que ele supera algoritmos RL estado-da-arte (livre de modelo e baseado em modelo), assim como métodos de meta-aprendizado estado-da-arte especialmente projetados para lidar com a não-estacionariedade.

Palavras-chave: Aprendizado por reforço. não-estacionariedade. RL baseado em modelo. detecção de change-point.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
MDP	Markov Decision Process
CPD	Change-Point Detection
FAR	False Alarm Rate
LLR	Log-Likelihood Ratio
KL	Kullback-Leibler
CUSUM	Cumulative Sum
MCUSUM	Multivariate Cumulative Sum
MLP	Multi-Layer Perceptrons
ReLU	Rectified Linear Units
MPC	Model-Predictive Control
CEM	Cross-Entropy Method
MBCD	Model-Based RL Context Detection
SAC	Soft Actor-Critic
MBPO	Model-Based Policy Optimization
GrBAL	Gradient-Based Adaptive Learning
ReBAL	Recurrence-Based Adaptive Learning

CONTENTS

1 INTRODUCTION

Reinforcement learning (RL) techniques have been successfully applied to solve high-dimensional sequential decision problems. However, if the state transition and/or reward functions change unexpectedly, according to latent factors unobservable to the agent, the system is effectively tasked with optimizing behavior policies that maximize performance over a (possibly infinite) random sequence of Markov Decision Processes (MDPs). Each MDP is drawn from an unknown distribution and is henceforth referred to as a *context*. This is known as a non-stationary setting. Designing efficient algorithms to tackle this problem is a known challenge in RL (PADAKANDLA, 2020). The key difficulties here result from *(i)* the need to quickly and reliably detect when the underlying system dynamics has changed; and *(ii)* the need to effectively learn and deploy adaptable prediction models and policies, specialized in particular contexts, while allowing the agent to (when appropriate) reuse previously-acquired knowledge. Non-stationary environments often result from systems whose dynamics are inherently time-dependent; from agents that are tasked with learning policies under noisy or missing sensors; or from problems where the agent faces sequences of unlabeled/unidentified tasks, each one with its own transition dynamics and reward functions.

Non-stationary settings arise naturally in many situations. Humans, for instance, are capable of learning to solve sequences of tasks from few experiences while preserving knowledge from older experiences (LAKE; SALAKHUTDINOV; TENENBAUM, 2015). Consider, for example, a person realizing the need to adapt their gait after an accident, learning novel gait patterns to use when walking with crutches, and then, after a period of recovery, successfully re-deploying normal walking gaits. This corresponds to a non-stationary scenario where the agent needs to learn specialized dynamics models and policies for tackling different contexts/learning scenarios. We introduce an algorithm that efficiently learns decision-making strategies in this setting. It assumes an agent that experiences random sequences of contexts (MDPs) drawn from some unknown distribution, and it is capable of optimizing behaviors even when a pre-training phase (during which the agent interacts with sample contexts) is not available. The agent's goal is to rapidly deploy decision-making policies that are appropriate for each randomly-arriving context—even when the number of latent contexts is unknown and when the context distribution cannot be modeled nor controlled by the agent. We are particularly interested in the case where quick readaptation and sample efficiency are paramount to achieving good performance; for instance, in cases where collecting experiences and acquiring policies from scratch, when dealing with a novel context, is unfeasible.

Many existing related works tackle non-stationary problems either by detecting when the underlying MDP changes, or via meta-learning approaches that construct a prior model (or policy) capable of rapidly generalizing to novel random contexts. Hadoux et al., for example, introduced a technique based on change-point detection (CPD) algorithms to deal with non-stationary problems with discrete state spaces (HADOUX; BEYNIER; WENG, 2014; BANERJEE; LIU; HOW, 2017). We, by contrast, address the more general setting of high-dimensional continuous RL problems. Supervised meta-learning algorithms (FINN; ABBEEL; LEVINE, 2017) have also been recently combined with RL to enable fast adaptation under changing domains (NAGABANDI et al., 2019; NAGABANDI; FINN; LEVINE, 2019). Nagabandi et al. introduced a model-based algorithm where a meta-learning technique is used to construct probabilistic dynamics models capable of rapidly adapting to recent experiences—either by updating the hidden state of a recurrent neural network (DUAN et al., 2016), or by updating the model parameters via a small number of gradient steps (FINN; ABBEEL; LEVINE, 2017). Meta-learning methods typically assume disjoint training and testing phases, so that an agent can be pre-trained over randomly sampled contexts prior to its deployment. We, by contrast, do not require a pre-training phase. Meta-learning methods also typically assume that the distribution over contexts experienced during training is the same as the one experienced during testing, so that agents can adapt to novel environments with structural similarities to those previously experienced. We, by contrast, do not require that contexts are sampled from a previously-seen distribution, nor that contexts share structural similarities with previously-experienced ones.

To address these limitations, we introduce an algorithm that analyzes a possibly infinite stream of data and computes, in real-time, high-confidence change-point detection statistics that reflect whether novel, specialized policies need to be deployed to tackle new contexts, or whether a previously-optimized policy may be reused. We call our algorithm Model-Based RL Context Detection, or MBCD. We formally show that it minimizes the delay until unforeseen changes to a context are detected, thereby allowing for rapid responses, and that it allows for formal bounds on the rate of false alarm—which is of interest when minimizing the agent's regret over random sequences of contexts. Our method constructs a mixture model composed of a (possibly infinite) ensemble of probabilistic dynamics predictors that model the different modes of the distribution over underlying latent MDPs. Our method is capable of optimizing policies based on streams of arbitrarily different contexts, with unknown boundaries, and which may be drawn from an arbitrary, unknown distribution.

We evaluate our algorithm on high-dimensional continuous reinforcement learning tasks and empirically show *(i)* that state-of-the-art reinforcement learning algorithms struggle to deal with non-stationarity; and *(ii)* that our method outperforms state-of-the-art meta-learning methods specifically tailored to deal with non-stationary environments—in particular, when the agent is faced with MDPs that are off-distribution with respect to the set of training contexts provided to the agent, or when novel contexts are structurally different from previously-observed ones.

This work is organized as follow: Chapter 2 presents theoretical background of RL and model-based RL; Chapter 3 describes related works on non-stationary RL; Chapter 4 defines our problem formulation and presents a change-point detection approach to this problem; Chapter 5 introduces our proposed algorithm; Chapter 6 presents the experimental settings and results, and Chapter 7 the conclusions.

2 THEORETICAL BACKGROUND

2.1 Reinforcement Learning

Reinforcement Learning (RL) is the field of Artificial Intelligence in which an agent learns to maximize a reward signal while interacting with an environment (SUTTON; BARTO, 2018). In contrast to supervised learning, it does not need a large database of pre-labeled training data. This opens up many applications for machine learning for which no such database exists. In recent years, RL has achieved impressive results in a broad range of fields, as for instance: robotics tasks (HAARNOJA et al., 2018), video games (VINYALS et al., 2019), traffic systems (ZIEMKE; ALEGRE; BAZZAN, 2020), healthcare (YU; LIU; NEMATI, 2019), etc. Many of the recent advances in the field were possible due to the combination of RL algorithms with deep neural networks as function approximators, from which emerged the field of Deep Reinforcement Learning (DRL).

An RL problem is typically formulated as a Markov Decision Process (MDP). A MDP $\mathcal{M}$ is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, d^0)$, where $\mathcal{S}$ is a (possibly continuous) state space, $\mathcal{A}$ is a (possibly continuous) action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function specifying the distribution over next states, given the current state and action, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\gamma \in [0, 1]$ is the discount factor, and d^0 is an initial state distribution. In what follows, S_t, A_t, and R_t are the random variables corresponding to the state, action, and reward at time step t.

A policy π is a mapping from each state, $s \in \mathcal{S}$, and action, $a \in \mathcal{A}$, to the probability $\pi(a|s)$ of taking action a when in state s. The value of a state s under a policy π, denoted by $v_\pi(s)$, is the expected sum of rewards obtained when starting in s and following π thereafter, under a (possibly infinite) horizon H:

$$v_\pi(s) = \mathbb{E}\left[\sum_{j=0}^{H-1} \gamma^j R_{t+j} | S_t = s, \pi, \mathcal{M}\right]. \tag{2.1}$$

Similarly, we define the action-value function, denoted $q_\pi(s, a)$, as the expected return of taking action a in state s, and thereafter following policy π:

$$q_\pi(s, a) = \mathbb{E}\left[\sum_{j=0}^{H-1} \gamma^j R_{t+j} | S_t = s, A_t = a, \pi, \mathcal{M}\right]. \tag{2.2}$$

When the optimal action-values q^* are known/learned, an optimal policy π^* can be simply defined as:

$$\pi^*(s) = \underset{a}{\text{argmax}}\ q^*(s,a). \tag{2.3}$$

RL algorithms are commonly divided into two categories: model-free and model-based methods. Model-free methods try to directly learn a policy and/or value function from observed data, without any prior/learned information regarding the dynamics of the MDP. Model-based methods, instead, aim to increase the sample efficiency, i.e. the number of environment interactions needed in order to learn a policy, by learning a predictive model of the MDP dynamics. The learned model of the environment can be used to generate additional samples used to train the policy (SUTTON; BARTO, 2018) or to derive a controller (CAMACHO; ALBA, 2013). Although model-free algorithms have achieved remarkable success in many settings, they need a large number of samples (environment interactions), which limits them mostly to simulators.

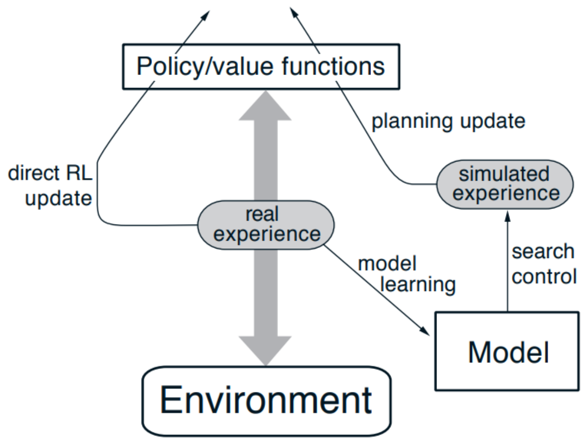
2.2 Model-Based Reinforcement Learning

Model-based RL algorithms aim to increase the sample efficiency, i.e. the number of environment interactions needed in order to learn a policy, by learning a model of the MDP dynamics (SUTTON; BARTO, 2018). In model-based RL (MOERLAND; BROEKENS; JONKER, 2020), the agent typically learns a model of the state-transition function $\mathcal{T}$ and/or the reward function $\mathcal{R}$ using the experienced transitions collected by interacting with the environment.

In the case where the RL agent learns a model $p(S_{t+1}, R_t|S_t, A_t)$, which predicts the next state and reward given the current state and action, it can use this model in combination with a model-free algorithm to accelerate the learning of a policy. The original proposal of such a combination comes from the Dyna algorithm (SUTTON, 1990), which alternates between model learning, data generation under a model, and policy learning using the model data (see Figure 2.1).

Another class of model-based RL algorithms uses the learned dynamics models for planning using a Model Predictive Control (MPC) method (GARCíA; PRETT; MORARI, 1989). MPC methods work by generating candidate action sequences from a distribution, and evaluating each candidate sequence using the learned dynamics. The optimal action sequence is approximated as the one with the highest return. The MPC agent only applies the first action from the optimal sequence and re-plans at every time-step. This planning

Figure 2.1: The general Dyna architecture. Image from (SUTTON; BARTO, 2018).



process can be further improved using the Cross-Entropy Method (CEM) (BOTEV et al., 2013). When using CEM, candidate action sequences are iteratively sampled from a candidate Gaussian distribution, which has its mean adjusted based on best performing action samples. Then, the mean of the adjusted candidate distribution is used as action and re-plan is done at every time step.

The success of the model-based approach hinges critically on the quality of the predictions of the dynamics model. In literature, there are many model architecture choices. Linear models (SUTTON et al., 2008) and Gaussian processes (DEISENROTH; RASMUSSEN, 2011) provide good performance in the low-data regime. Neural network predictive models (NAGABANDI et al., 2018), in contrast, require more data samples, but provide accurate predictions even in domains with high-dimensional state and action spaces. Recently, works have shown that ensembles are effective in preventing a policy or a controller from exploiting the inaccuracies of any single model (CHUA et al., 2018; JANNER et al., 2019).

3 RELATED WORK

In this chapter we review the most relevant related works which proposed algorithms to deal with non-stationary RL domains. In Table 3.1, we summarize the described methods and compare them to our proposed method. Notice that our method is the only model-based method able to deal with continuous state-spaces without a pre-training period.

3.1 Context Detection Methods

Dealing with non-stationary environments in RL via context change-point detection has been studied in discrete state and action spaces settings.

Silva et al. introduced Reinforcement Learning Context Detection (RLCD) (SILVA et al., 2006). RLCD is a model-based algorithm that estimates the state transition and reward functions from collected samples, while quality measures are used to choose and update the current model. If the maximum quality is below a given threshold, a new model is added to the list of known models, uniformly initialized and selected as the next current model. RLCD does not require *a priori* knowledge about the number of environment contexts nor a pre-training phase (like meta-learning algorithms). However, it is only applicable to purely discrete settings.

Basso et al. introduced an algorithm for solving reinforcement learning problems in non-stationary continuous time and state environments through context detection (BASSO; ENGEL, 2009). It computes the instantaneous quality of a model as a value inversely proportional to its prediction error. A quality trace integrates the instantaneous quality over the time and, at each moment, the model with the highest quality trace is chosen as the current active model. If the quality traces of all models are worse than the minimum allowed, the system assumes that the environment is in a new context. However, the dynamics models are estimated using linear approximation functions, which limits the algorithm to domains in which the dynamics can be linearly approximated. Our method, by contrast, uses probabilistic neural networks, which are powerful non-linear function approximators more suitable to tasks with complex dynamics, such as robotics locomotion. In addition, unlike MBCD, their method can not perceive non-stationarity in the reward function.

Hadoux et al. introduced an extension of RLCD that replaces its quality measures with a CUSUM-based method to perform change-point detection (HADOUX; BEYNIER; WENG, 2014). Similar to RLCD, this method estimates the transition and reward functions for all contexts. Each time step, a CUSUM statistic is computed for each known context, indicating whether the current context has changed. An additional CUSUM statistic is also computed for a fixed transition function represented by a uniform model (one which gives equal probability of transition between all states for all actions). If this fixed model is the most likely to have generated the experienced transitions, a new model is instantiated. Unlike our method, however, it is only applicable to discrete state settings.

Banerjee et al. shows that in full information case, i.e., when complete model information is known, the change detection approach of (HADOUX; BEYNIER; WENG, 2014) leads to loss in performance with delayed detection. Based on this observation, in (BANERJEE; LIU; HOW, 2017) it is designed a two-threshold policy switching method based on KL divergence between transition models in order to rapidly detect context changes. This is a principled method but—unlike MBCD—requires prior knowledge of the dynamics model of all contexts.

3.2 Model-Free Methods

Model-free RL algorithms try to deal with non-stationarity without modeling the MDP dynamics. Instead, they focus on building policies resilient to catastrophic forgetting and capable of fast adaptation.

In (ZHOU et al., 2019), it is proposed a multi-level model architecture, named as Policy Residual Representation (PRR), as well as a training method that enables a single model to represent multiple levels of experience. In each level of PRR, there are one or more component modules, corresponding to a subset of the contexts. The training starts from the top level with one module corresponding to all the contexts, i.e., the module is the average policy over all the contexts. In each following level, a module is learned over a selected subset of contexts according to a predefined mask. Moreover, when learning the module, all the upper levels are fixed, and thus the module learns a residual policy over the selected contexts. In this way, PRR forms a multilevel architecture, where the experience of different granularities can be represented.

In (KAPLANIS; SHANAHAN; CLOPATH, 2019), it is developed an approach called Policy Consolidation (PC) to mitigate catastrophic forgetting. Policy consolidation means that the current behavioural policy is distilled into a cascade of hidden neural networks that record policies at multiple timescales. Rather than relying on task boundaries, consolidation in occurs at all times, with the agent's policy being continually distilled into a cascade of hidden networks that evolve over a range of timescales. The hidden networks, in turn, distill knowledge back through the cascade into the policy network in order to ensure that the agent's policy does not deviate too much from where it was previously. The policies are encoded by the parameters of the neural network and the distance between the parameters of two such networks can be used as a substitute for the distance between policies (represented by the networks). This substitute measure is also incorporated in the loss function used for training the policy network.

In (ROLNICK et al., 2019), Continual Learning with Experience And Replay (CLEAR) is introduced. CLEAR mixes on-policy learning from novel experiences (for plasticity) and off-policy learning from replay experiences (for stability). For additional stability, they introduce behavioral cloning between the current policy and its past self. Their method outperforms other previous methods that assumes the boundaries between tasks are known in a set of Atari tasks. However, it is not expected that it could work well in continuous control tasks where the dynamics changes significantly and reusing experiences from past tasks possibly introduces negative transfer.

3.3 Meta-Learning Methods

Meta-learning algorithms are used to train a prior over dynamics models that can, when combined with recent data, be rapidly adapted to novel contexts.

In (NAGABANDI et al., 2019), two model-based meta-learning algorithms are proposed: Gradient-Based Adaptive Learner (GrBAL) and Recurrence-Based Adaptive Learner (ReBAL). GrBAL employs the Model-Agnostic Meta-Learning (MAML) method (FINN; ABBEEL; LEVINE, 2017) to learn the parameters of a meta-learning prior over the dynamics model, given a set of training contexts. This prior is constructed so that it serves as a good initial model for any new contexts that the agent encounters after a pre-training phase. After training, thus, such a meta-learned dynamics model is capable of quickly adapting to a current task's dynamics by taking only a few gradient steps. ReBAL works similarly to GrBAL, but instead of taking gradient steps to adapt a prior model to novel contexts, it uses a recurrent neural network that learns its own update rule (vs. a gradient update rule) through its hidden state. Both ReBAL and GrBAL use MPC for selecting actions by planning for a certain horizon using the adapted dynamics model.

In (NAGABANDI; FINN; LEVINE, 2019), a similar approach is used. However, the task changes are seen as a Dirichlet process and have their priors modeled with a Chinese restaurant process controlled by a parameter α. The higher the value of α, the more predisposed the method is to instantiate a new task after t time steps.

These methods, unlike MBCD, were designed to tackle settings where the non-stationarity solely results from changes to the dynamics, but not to the agent's goals/reward function. Reward functions are assumed to be known *a priori*. These methods also require an explicit pre-training train phase, prior to deployment, and assume that the distribution of training and testing contexts is the same. Our method, by contrast, is better suited to continual online settings where a pre-training phase is not possible, and where the agent is tasked with dealing with streams of arbitrarily different contexts with unknown boundaries.

Additionally, meta-learning algorithms need all tasks to share a common structure that can be exploited for fast learning. Hence, although meta-learning is attractive for reusing previous knowledge to speed the learning of new tasks, their underlying assumptions are not realistic for many real-world scenarios in which the world changes abruptly in an unpredictable manner.

Table 3.1: Summary of related works.

Method	*State space*	*Number of tasks*	*MDP dynamics*	*Pre-training*	*Non-stationarity*	*Policy*
(SILVA et al., 2006) and (HADOUX; BEYNIER; WENG, 2014)	Discrete	Unlimited	Learned	No	Dynamics and Reward	Learned (model-based)
(BASSO; ENGEL, 2009)	Continuous	Unlimited	Learned (linear models)	No	Dynamics	Learned (model-based)
(BANERJEE; LIU; HOW, 2017)	Discrete	Known *a priori*	Known *a priori*	No	Dynamics	Known *a priori*
(ZHOU et al., 2019)	Continuous	Unlimited	Model-free	Yes	Dynamics and Reward	Learned (model-free)
(KAPLANIS; SHANAHAN; CLOPATH, 2019)	Continuous	Unlimited	Model-free	No	Dynamics	Learned (model-free)
(ROLNICK et al., 2019)	Image Frames	Unlimited	Model-free	No	Dynamics and Reward	Learned (model-free)
(NAGABANDI et al., 2019)	Continuous	Unlimited	Learned	Meta-learning*	Dynamics	MPC
(NAGABANDI; FINN; LEVINE, 2019)	Continuous	Unlimited	Learned	Meta-learning*	Dynamics	MPC
Ours	Continuous	Unlimited	Learned	No	Dynamics and Reward	Learned (model-based)

*These methods require an explicit pre-training train phase, and assume that the distribution of training and testing contexts is the same.

4 NON-STATIONARITY AND CHANGE-POINT DETECTION

In this Chapter, we formulate the problem of learning in non-stationary environments through RL (Section 4.1). Then, in Section 4.2 we demonstrate how this problem can be tackled via a change-point detection approach and present the theoretical guarantees that such methods introduce. Next, in Chapter 5 we introduce an algorithm that effectively implements these ideas under the assumptions discussed in this Chapter.

4.1 Problem Formulation

We define a non-stationary environment as a family of MDPs $\{\mathcal{M}_z\}_{z\in\mathbb{N}^+}$. Each MDP $\mathcal{M}_z$ is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}_z, \mathcal{R}_z, \gamma, d^0)$, as defined in Section 2.1. We assume that the agent observes a random sequence $(\mathcal{M}_0, \mathcal{M}_1, \ldots)$ of MDPs—called *contexts*—drawn independently from some unknown distribution. We assume that the number of contexts, $|\{\mathcal{M}_z\}|$, is unknown. These definitions are similar to those discussed in (BANERJEE; LIU; HOW, 2017) and (PADAKANDLA, 2020). Let z be a latent index variable indicating a particular MDP, $\mathcal{M}_z$. We assume that each MDP's transition and reward function are parameterized by a latent vector θ_z. Let $p_{\theta_z}(S_{t+1}, R_t|S_t, A_t)$ denote the joint conditional probability distribution over next-state and reward associated with MDP $\mathcal{M}_z$. We do not impose any smoothness assumptions on how variations to θ_z affect $\mathcal{T}_z$ and $\mathcal{R}_z$: contexts may be arbitrarily different and share no structural similarities.

Let the time steps in which context changes occur be an increasing sequence of integer random variables, $\{C_i\}_{i\geq 1}$, for which a prior $\phi(C_i)$ is unknown or cannot be defined. We call each C_i a *change-point*. At every change-point C_i, the current context $\mathcal{M}_z$ is replaced by a random MDP drawn from $\{\mathcal{M}_z\}$. To perform well, an agent must rapidly detect changes to its environment and deploy an appropriate policy. If a new random context differs significantly from previously-experienced ones, the agent may have to learn a policy from scratch; otherwise, it may choose to reuse previously-acquired knowledge to accelerate learning and avoid catastrophic forgetting.

At each time t, when interacting with its environment, the agent selects an action A_t based on its state S_t according to a stochastic policy $\pi(\cdot|S_t)$. Let $v^H_{\pi,\mathcal{M}_z}$ be the value function associated with policy π, MDP $\mathcal{M}_z$, and defined over a horizon H:

$$v^H_{\pi,\mathcal{M}_z}(s) = \mathbb{E}\left[\sum_{j=0}^{H-1} \gamma^j R_{t+j} | S_t = s, \pi, \mathcal{M}_z\right]. \tag{4.1}$$

To simplify our analysis, we first consider (without loss of generality) the simpler case of a family of MPDs $\{\mathcal{M}_0, \mathcal{M}_1\}$, where a context change occurs from $\mathcal{M}_0$ to $\mathcal{M}_1$ at some unobserved random time C. The mathematical arguments that follow can be extended to the more general setting with an arbitrary number of contexts. Let Π^* be a policy that follows the optimal policy for $\mathcal{M}_0$, π_0, before C, and the optimal policy for $\mathcal{M}_1$, π_1, afterwards. This policy's value function is defined as:

$$v^\infty_{\Pi^*,\{\mathcal{M}_0,\mathcal{M}_1\}}(s) = \mathbb{E}\left[v^C_{\pi_0,\mathcal{M}_0}(s) + \gamma^C \mathbb{E}_{s' \sim \rho^{\pi_0}_{\mathcal{M}_0,C-1}}\left[v^\infty_{\pi_1,\mathcal{M}_1}(s')\right]\right], \tag{4.2}$$

where $\rho^\pi_{\mathcal{M},t}$ denotes the distribution of states reachable after following policy π for t steps under MDP $\mathcal{M}$. Notice that Eq. 4.2 models the case when $\mathcal{M}_1$ starts in the (random) state where $\mathcal{M}_0$ terminated, immediately prior to the random time C. This implies that $d^0_{\mathcal{M}_1} = \rho^{\pi_0}_{\mathcal{M}_0,C-1}$.

By contrast, consider an alternative policy Π that follows policy π_1 only after a random time Γ, for $\Gamma > C$; that is, a policy that deploys the correct decision-making strategy for $\mathcal{M}_1$ with a delay of $\Delta = \Gamma - C$ steps. Its value function is given by:

$$\begin{aligned} v^\infty_{\Pi,\{\mathcal{M}_0,\mathcal{M}_1\}}(s) = \mathbb{E}[v^C_{\pi_0,\mathcal{M}_0}(s) + \\ \gamma^C \mathbb{E}_{s' \sim \rho^{\pi_0}_{\mathcal{M}_0,C-1}}\left[v^\Delta_{\pi_0,\mathcal{M}_1}(s')\right] + \\ \gamma^{C+\Delta+1} \mathbb{E}_{s' \sim \rho^{\pi_0}_{\mathcal{M}_1,\Delta}}\left[v^\infty_{\pi_1,\mathcal{M}_1}(s')\right]]. \end{aligned} \tag{4.3}$$

Notice that we can rewrite Eq. 4.2 in the same form as Eq. 4.3:

$$\begin{aligned} v^\infty_{\Pi^*,\{\mathcal{M}_0,\mathcal{M}_1\}}(s) = \mathbb{E}[v^C_{\pi_0,\mathcal{M}_0}(s) + \\ \gamma^C \mathbb{E}_{s' \sim \rho^{\pi_0}_{\mathcal{M}_0,C-1}}\left[v^\Delta_{\pi_1,\mathcal{M}_1}(s')\right] + \\ \gamma^{C+\Delta+1} \mathbb{E}_{s' \sim \rho^{\pi_1}_{\mathcal{M}_1,\Delta}}\left[v^\infty_{\pi_1,\mathcal{M}_1}(s')\right]]. \end{aligned} \tag{4.4}$$

We now define the regret $\mathcal{L}(\Delta)$ as the expected discounted sum of rewards lost due to the delay $\Delta = \Gamma - C$, when changing from π_0 to π_1 only at time step Γ. This quantity is given by difference between Eq. 4.3 and Eq. 4.4. Since we are interested in minimizing the delay Δ, we assume the adversarial case when policies π_0 and π_1 do, in the short-term (i.e., within the delay window Δ), have a nearly indistinguishable distribution over the states that are reachable in Δ steps. In particular, we assume that the KL divergence between $\rho^{\pi_0}_{\mathcal{M}_1,\Delta}$ and $\rho^{\pi_1}_{\mathcal{M}_1,\Delta}$ is bounded and small for small values of Δ. In this case, the regret can be approximated by:

$$\mathcal{L}(\Delta) \approx \mathbb{E}\left[\gamma^C \mathbb{E}_{s' \sim \rho^{\pi_0}_{\mathcal{M}_0,\Gamma-1}}\left[v^{\Delta}_{\pi_1,\mathcal{M}_1}(s') - v^{\Delta}_{\pi_0,\mathcal{M}_1}(s')\right]\right] \tag{4.5}$$

as $\mathcal{D}_{KL}(\rho^{\pi_0}_{\mathcal{M}_1,\Delta} || \rho^{\pi_1}_{\mathcal{M}_1,\Delta}) \to 0$. From Eq. 4.5, it should be clear that to maximize the expected return over the random sequence of MDPs, one needs to minimize regret; and to minimize regret, one needs to minimize the random delay Δ. This definition can be extended in a straightforward way to the case where there is a sequence of random change-points, $\{C_i\}_{i\geq 1}$. In particular, the regret will be defined in terms not of a single random delay, but of a sequence of random delays associated with the corresponding random contexts that are observed by the agent.

In the next chapter, we introduce a method capable of minimizing the worst-case delay until unforeseen changes to a stochastic process are detected, while also bounding the rate of false alarm—i.e., the likelihood that the method will *incorrectly* indicate that a context change occurred.

4.2 High-Confidence Change-Point Detection

Change-point detection (CPD) algorithms (VEERAVALLI; BANERJEE, 2012; AMINIKHANGHAHI; COOK, 2016) are designed to detect whether (and when) a change occurs in the distribution generating random observations from an arbitrary stochastic process. These methods have been widely used in a variety of fields—from financial markets (LAM; YAM, 1997) to biomedical signal processing (SIBANDA; SIBANDA, 2007). Although CPDs have been applied to reinforcement learning problems (HADOUX; BEYNIER; WENG, 2014; BANERJEE; LIU; HOW, 2017), the application and formal analysis of such methods has been restricted to discrete state spaces settings.

In the *online* CPD setting, a sequential detection procedure is defined with the objective of rapidly and reliably estimating when the parameter θ of some underlying distribution or stochastic process has changed. Online CPD algorithms should produce high-confidence estimates, Γ, of the true change-point time, C. Notice that Γ is a random variable whose stochasticity results from the unknown stochastic prior over context changes, ϕ, and from the fact that each MDP in $\{\mathcal{M}_z\}$ produces random trajectories of states, actions, and rewards.

Suppose that at each time t, while the agent interacts with $\mathcal{M}_0$, sample next-state and rewards are drawn from $p_{\theta_0}(S_{t+1}, R_t|S_t, A_t)$, where θ_0 is the latent vector parameterizing $\mathcal{M}_0$'s transition and reward functions. At some unknown random change-point C, the context changes to $\mathcal{M}_1$, and experiences that follow are drawn from $p_{\theta_1}(S_{t+1}, R_t|S_t, A_t)$. We propose to identify such a change by computing high-confidence statistics that reflect whether θ_0 has changed. This can be achieved by introducing a minimax formulation of the CPD problem, as discussed by Pollak (POLLAK, 1985). In this formulation, the goal of a CPD algorithm is to compute a random estimate, Γ, of the latent change-point time C, such that *(i)* it minimizes the worst case expected detection delay, $\Delta_{worst}(\Gamma)$, associated with the random estimates Γ produced by a particular CPD algorithm, when considering all possible change-points C; and *(ii)* bounds on the maximum false alarm rate (FAR) may be imposed. The worst-case expected detection delay, $\Delta_{worst}(\Gamma)$, and the FAR, are defined as:

$$\Delta_{worst}(\Gamma) = \sup_{c\geq 1} \mathbb{E}[\Gamma - C|\Gamma \geq C, C = c], \tag{4.6}$$

$$\text{FAR}(\Gamma) = \frac{1}{\mathbb{E}[\Gamma|C = \infty]}, \tag{4.7}$$

where the expectations in Eq. 4.6 and Eq. 4.7 are over the possible histories of experiences produced by the stochastic process, and where conditioning on $C = \infty$ indicates the random event where the context never changes. Given these definitions, the objective of a high-confidence change-point detection process is the following:

$$\inf_{\Gamma} \Delta_{worst}(\Gamma) \text{ subject to } \text{FAR}(\Gamma) \leq \alpha, \tag{4.8}$$

where α denotes the desired upper-bound on the false alarm rate.

When θ_0 and θ_1 are known, the Log-Likelihood Ratio (LLR) statistic can be used to recursively compute the Cumulative Sum (CUSUM) statistic (PAGE, 1954). As we will discuss next, such a statistic can be used to construct a high-confidence change-point detection method. The LLR statistic, L_t, and the CUSUM statistic, W_t, are updated at each time t as follows:

$$L_t = \log \frac{p_{\theta_1}(S_{t+1}, R_t|S_t, A_t)}{p_{\theta_0}(S_{t+1}, R_t|S_t, A_t)}, \tag{4.9}$$

$$W_t = \max\left(0, W_{t-1} + L_t\right),\ W_0 = 0. \tag{4.10}$$

Importantly, notice that before the change-point C is reached, $\mathbb{E}[L_t] < 0$, which implies that the expected value of W_t is zero. After the change-point C is reached, $\mathbb{E}[L_t] > 0$, and therefore W_t will tend to increase. Higher values of W_t, thus, serve as principled statistics reflecting evidence that a change-point has occurred between θ_0 to θ_1. The random time Γ when a change-point is estimated to have happened is defined as the first time when the CUSUM metric W_t becomes greater than a detection threshold h:

$$\Gamma = \min\{t \geq 1 : W_t > h\}. \tag{4.11}$$

(LORDEN, 1971), shows that choosing $h = |\log \alpha|$ ensures that $\mathrm{FAR}(\Gamma) \leq \alpha$. Furthermore, (LAI, 1998) demonstrated that the CUSUM detection time Γ is asymptotically optimum with respect to the problem specified in Eq. 4.6. In particular, they showed that the worst expected detection delay (under $h = |\log \alpha|$) respects the following approximation:

$$\Delta_{worst}(\Gamma) \approx \frac{|\log \alpha|}{\mathcal{D}_{KL}(p_{\theta_1}||p_{\theta_0})} \text{ as } \alpha \to 0. \tag{4.12}$$

In Eq. 4.12, the denominator $\mathcal{D}_{KL}(p_{\theta_1}||p_{\theta_0}) = \mathbb{E}_{\theta_1}\left[\log \frac{p_{\theta_1}(S_{t+1},R_t|S_t,A_t)}{p_{\theta_0}(S_{t+1},R_t|S_t,A_t)}\right]$ is the Kullback–Leibler divergence under θ_1. Eq. 4.12 implies that the larger the difference between the distributions p_{θ_1} and p_{θ_0}, the smaller the expected delay ($\Delta = \Gamma - C$) for detecting a change-point. The above results allow us to construct high-confidence statistics reflecting whether (and when) a context has changed; importantly, they are both accurate and have bounded false alarm rate.

5 MODEL-BASED RL CONTEXT DETECTION

In this chapter, we introduce an algorithm that iteratively applies a CUSUM-related procedure to detect context changes under the assumptions discussed in Chapter 4. The algorithm incrementally builds a library of models and policies for tackling arbitrarily different types of contexts; i.e., contexts that may result from quantitatively and qualitatively different underlying causes for non-stationarity—ranging from unpredictable environmental changes (such as random wind) to robot malfunctions. Our method can rapidly deploy previously-constructed policies whenever contexts approximately re-occur, or learn new decision-making strategies whenever novel contexts, with no structural similarities with respect to previously-observed ones, are first encountered. Unlike existing approaches (see Chapter 3), our method is capable of *(i)* optimizing policies *online*; *(ii)* without requiring a pre-training phase; *(iii)* based on streams of arbitrarily different contexts, with unknown change-point boundaries; and *(iv)* such that contexts may be drawn from an arbitrary, unknown distribution.

5.1 Method Overview

We now introduce a *high-level* description of our method (Model-Based RL Context Detection, or MBCD). In subsequent sections, we provide details for each of the method's components. As the agent interacts with a non-stationary environment, context changes are identified via a multivariate variant of CUSUM (HEALY, 1987), called MCUSUM. MCUSUM-based statistics inherit the same formal properties as those presented in Section 4.2. In particular, they formally guarantee that MBCD can detect context changes with minimum expected delay, while simultaneously bounding the false alarm rate. As a consequence, MBCD can effectively identify novel environmental dynamics while ensuring, with high probability, that new context-specific policies will only be construct when necessary.

As new contexts are identified by this procedure, MBCD updates a mixture model, M, composed of a (possibly infinite) ensemble of probabilistic context dynamics predictors, whose purpose is to model the different modes of the distribution over underlying latent MDPs/contexts. New models are added to the ensemble as qualitatively different contexts are first encountered. The mixture model M associates, with each identified context $\mathcal{M}_z$, a learned joint distribution p_{θ_z} over next-state dynamics and rewards associated.

Let K be the number of context models currently in the mixture. After each agent experience, MBCD identifies the most likely context, z_t, by analyzing a set of incrementally-estimated MCUSUM statistics (see Section 5.3). Whenever a novel context—one with dynamics that are qualitatively different from those previously-experienced—is observed, a new model is added to the mixture. Context-specific policies, π_{ψ_z}, are trained via a Dyna-style approach based on the corresponding learned joint prediction model of $\mathcal{M}_z$, p_{θ_z} (see Section 5.4). We provide details in the next sections. Pseudocode for MBCD is shown in Algorithm 1.

Algorithm 1: Model-Based RL Context Detection

```
Input: Non-stationary environment $E$, threshold $h$.
1  $z_0 \leftarrow 1$; $K \leftarrow 1$; $W_{z_0,0} \leftarrow 0$; $W_{\text{new},0} \leftarrow 0$
2  Initialize model $p_{\theta_{z_0}}$, policy $\pi_{\psi_{z_0}}$, datasets $D_{z_0}$ and $D_{model}$
3  $M \leftarrow \{p_{\theta_{z_0}}\}$
4  for $t = 0 \ldots \infty$ do
5      Execute action $a_t \sim \pi_{\psi_{z_t}}$, observe $s_{t+1}, r_t$
6      Update MCUSUM statistics $W_{k,t}, \forall k$, with Eq. 5.7
7      Update $z_t$ with Eq. 5.10
8      if $z_t \neq z_{t-1}$ then // Context changed
9          Reset MCUSUM statistics
10         $D_{model} = \{\}$
11         if $z_t = new$ then // New context detected
12             $K \leftarrow K + 1$; $z_t \leftarrow K$
13             Initialize $p_{\theta_{z_t}}$ and $D_{z_t}$
14             Let $\pi_{\psi_{z_t}} \leftarrow \pi_{\psi_{z_{t-1}}}$
15             $M \leftarrow M \cup \{p_{\theta_{z_t}}\}$
16     $D_{z_t} \leftarrow D_{z_t} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
17     if $t \bmod F = 0$ then
18         $\theta_{z_t} \leftarrow \theta_{z_t} - \lambda_p \nabla \mathcal{J}_p(\theta_{z_t}, D_{z_t})$    // Update model
19         for $L$ simulated 1-step rollouts do
20             Sample state $s_i$ from tuples in $D_{z_t}$
21             $a_i \sim \pi_{\psi_{z_t}}(\cdot|s_i)$
22             $(s'_i, r_i) \sim p_{\theta_{z_t}}(\cdot|s_i, a_i)$
23             $D_{model} \leftarrow D_{model} \cup (s_i, a_i, r_i, s'_i)$
24     $\psi_{z_t} \leftarrow \psi_{z_t} - \lambda_\pi \nabla \mathcal{J}_\pi(\psi_{z_t}, D_{model} \cup D_{z_t})$    // Update policy
```

5.2 Stochastic Mixture Model of Dynamics

In this paper we assume that p_{θ_z}, the joint distribution over next-state dynamics and rewards associated with context $\mathcal{M}_z$, can be approximated by a multivariate Gaussian distribution. In particular, following recent work on model-based RL (JANNER et al., 2019; CHUA et al., 2018), MBCD models the dynamics of a given environment $\mathcal{M}_z$, $p_{\theta_z}(S_{t+1}, R_t|S_t, A_t)$, via a bootstrap ensemble of probabilistic neural networks whose outputs parameterize a multivariate Gaussian distribution with diagonal covariance matrix. The bootstrapping procedure accounts for epistemic uncertainty (i.e. uncertainty about model parameters), which is crucial when making predictions about the agent's dynamics in regions of the state space where experiences are scarce. For each context $\mathcal{M}_z$ identified by MBCD, an ensemble of N stochastic neural networks is instantiated and added to the mixture model M. Each network n in the ensemble is parameterized by θ_z^n and computes a probability distribution, $p_{\theta_z^n}$, that approximates p_{θ_z} by predicting the mean and covariance over next-state and rewards conditioned on the current state and action:

$$p_{\theta_z^n}(S_{t+1}, R_t|S_t, A_t) = \mathcal{N}(\mu_{\theta_z^n}(S_t, A_t), \Sigma_{\theta_z^n}(S_t, A_t)), \tag{5.1}$$

where $\mu_{\theta_z^n}(S_t, A_t)$ and $\Sigma_{\theta_z^n}(S_t, A_t)$ are the network outputs given input (S_t, A_t). To simplify notation, let $\mathbf{X}_t = (S_t, A_t) \in \mathbb{R}^{\dim(\mathcal{S})+\dim(\mathcal{A})}$ and $\mathbf{Y}_t = (S_{t+1}, R_t) \in \mathbb{R}^{\dim(\mathcal{S})+1}$. We follow (LAKSHMINARAYANAN; PRITZEL; BLUNDELL, 2017) and model the ensemble prediction as a Gaussian distribution whose mean and covariance are computed based on the mean and covariances of each component of the ensemble. In particular, the ensemble predictive model, $\hat{p}_{\theta_z}$, associated with a given context $\mathcal{M}_z$, is defined as:

$$\hat{p}_{\theta_z}(\mathbf{Y}_t|\mathbf{X}_t) = \mathcal{N}(\mu_z^*(\mathbf{X}_t), \Sigma_z^*(\mathbf{X}_t)), \tag{5.2}$$

where

$$\mu_z^*(\mathbf{X}_t) = N^{-1}\sum_{n=1}^{N}(\mu_{\theta_z^n}(\mathbf{X}_t)), \text{ and} \tag{5.3}$$

$$\Sigma_z^*(\mathbf{X}_t) = N^{-1}\sum_{n=1}^{N}\left(\text{diag}(\Sigma_{\theta_z^n}(\mathbf{X}_t)) + \mu_{\theta_z^n}^2(\mathbf{X}_t)\right) - \mu_*^2(\mathbf{X}_t). \tag{5.4}$$

MBCD stores all experiences collected while in context $\mathcal{M}_z$ in a buffer D_z. After every F steps, it uses data in D_z to update the ensemble model $\hat{p}_{\theta_z}$ by minimizing the negative log prediction likelihood loss function:

$$\mathcal{J}_p(\theta, D) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim D}[-\log p_\theta(s_{t+1}, r_t|s_t, a_t)]. \tag{5.5}$$

5.3 Online Context Change-Point Detection

As previously discussed, MBCD employs a multivariate variant of CUSUM, MCUSUM, to detect context changes with high confidence. (HEALY, 1987) demonstrated that, when detecting shifts in the mean of a multivariate Gaussian distribution, MCUSUM inherits all theoretical optimality guarantees possessed by the univariate CUSUM procedure. Furthermore, Healy also proved that the detection delay is independent of the dimensionality of the data.

In the particular case where the dynamics of each context are modeled as multivariate Gaussians, the LLR statistic can be computed as follows. To simplify notation, let $\mu_0 = \mu_{\theta_0}(S_t, A_t)$ and $\Sigma_0 = \Sigma_{\theta_0}(S_t, A_t)$. It is then possible to show that the LLR statistic, L_t, between distributions $p_{\theta_1}(\mathbf{Y}_t|\mathbf{X}_t)$ and $p_{\theta_0}(\mathbf{Y}_t|\mathbf{X}_t)$, is given by:

$$L_t = \log \frac{(2\pi)^{-\frac{d}{2}}|\Sigma_1|^{-\frac{1}{2}} \exp\{-0.5(\mathbf{Y}_t - \mu_1)\Sigma_1^{-1}(\mathbf{Y}_t - \mu_1)\}}{(2\pi)^{-\frac{d}{2}}|\Sigma_0|^{-\frac{1}{2}} \exp\{-0.5(\mathbf{Y}_t - \mu_0)\Sigma_0^{-1}(\mathbf{Y}_t - \mu_0)\}} \tag{5.6}$$

where d is the dimensionality of the multivariate Gaussian. At each time step t, MBCD uses L_t to compute MCUSUM statistics $W_{k,t}$ for each known context k, plus an additional statistic $W_{\text{new},t}$, used to infer, with high probability, whether a novel context has been first encountered:

$$W_{k,t} \leftarrow \max\left(0, W_{k,t-1} + \log \frac{p_{\theta_k}(\mathbf{Y}_t|\mathbf{X}_t)}{p_{\theta_{z_t}}(\mathbf{Y}_t|\mathbf{X}_t)}\right), \ \forall k \in [1, K] \cup [\text{new}]. \tag{5.7}$$

Here, $W_{\text{new},t}$ can be seen as evidence that a previously-unseen context has been first encountered, based on whether the likelihood of all known contexts $k \in [1, K]$ is smaller than $p_{\theta_{\text{new}}}$, where

$$p_{\theta_{\text{new}}}(\mathbf{Y}_t \,|\, \mathbf{X}_t) = \mathcal{N}(\hat{\mathbf{Y}}_t, \, \Sigma_{\theta_{z_t}}(\mathbf{X}_t)), \tag{5.8}$$

and

$$\hat{\mathbf{Y}}_t = \mathbf{Y}_t + \delta \, \text{diag}(\Sigma_{\theta_{z_t}}(\mathbf{X}_t)). \tag{5.9}$$

Intuitively, $W_{\text{new},t}$ indicates whether none of the known contexts is likely to have generated the observed transitions. In Eq. 5.8, $p_{\theta_{\text{new}}}$ is the likelihood of a new context under the alternative hypothesis that the true observation $\mathbf{Y}_t$ is δ standard deviations away from the true observation $\mathbf{Y}_t$. In particular, δ indicates the minimum meaningful change in the distribution's mean that we are interested in detecting. Given updated statistics $W_{k,t}$, the most likely current context, z_t (which may or may not have changed) can then be identified as:

$$z_t \leftarrow \begin{cases} \operatorname{argmax}_k W_{k,t}, & \text{if } \exists k \in [1, K] \cup [\text{new}] \text{ s.t. } W_{k,t} > h, \\ z_{t-1}, & \text{otherwise.} \end{cases} \tag{5.10}$$

If no alternative contexts are more likely to have generated the observations collected up to time t, no context change is detected and $z_t = z_{t-1}$.

Notice that in Eq. 5.7, models p_{θ_k} are assumed to be known *a priori*. In our setting, these models are estimated based on samples. MCUSUM has been studied in scenarios where the parameters of the distribution are known only approximately (MAHMOUD; MARAVELAKIS, 2013). In our work, we address this challenge by computing change-point detection statistics only after a small warm-up period within which the agent is allowed to operate in a given context. In particular, and similarly to Sekar et al. (SEKAR et al., 2020), we define the warm-up period by using ensemble disagreement as a proxy to quantify the system's uncertainty regarding the current distributions. Assuming a warm-up period where the agent is allowed to operate within each newly-encountered context is a common assumption in the area (NAGABANDI et al., 2019; NAGABANDI; FINN; LEVINE, 2019; RAKELLY et al., 2019). In fact, it is a *necessary* assumption: if contexts are allowed to change arbitrarily fast, adversarial settings can be constructed where all methods for dealing with non-stationary scenarios fail.

Finally, notice that a key element of Eq. 5.10 is the threshold h, against which each $W_{k,t}$ is compared in order to check if a context change has occurred. Different methods have been proposed to set h (SAHKI; GEGOUT-PETIT; WANTZ-MÉZIÈRES, 2020). Here, we take a conservative approach. As discussed in Section 4.2, setting $h = |\log \alpha|$ ensures that $FAR(\Gamma) \leq \alpha$. In this work, we set h by considering negligible values of α; e.g. $h = 100$ if $\alpha \approx 10^{-43}$. In our experiments, we observed that detection delays remain low even even for very conservative values of h.

5.4 Policy Optimization

Since MBCD estimates dynamics models for each context, it is natural to exploit such models to accelerate policy learning by deploying model-based RL algorithms. MBCD learns context-specific policies via a Dyna-style approach (SUTTON, 1990). In particular, at every time step t during which the model $p_{\theta_{z_t}}$ is trained, L 1-step simulated rollouts are sampled from $\pi_{\psi_{z_t}}$. Each rollout starts from a random state drawn from D_{z_t}. All rollouts are stored in a buffer, D_{model}.[1] Notice that this is similar to the procedure used by the Model-Based Policy Optimization (MBPO) algorithm (JANNER et al., 2019). Each context-specific policy $\pi_{\psi_{z_t}}$ is optimized by taking into account both real experiences (stored in D_{z_t}) and simulated experiences (stored in D_{model}).

Policy optimization is performed using the Soft Actor-Critic (SAC) algorithm (HAARNOJA et al., 2018). SAC alternates between a policy evaluation step, which estimates $q_\pi(s,a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t | S_t = s, A_t = a, \pi]$ using the Bellman backup operator, and a policy improvement step, which optimizes the policy π by minimizing the expected KL-divergence between the current policy and the exponential of a soft Q-function (HAARNOJA et al., 2018). Optimizing the policy, then, corresponds to minimizing the following loss function:

$$\mathcal{J}_\pi(\psi, D) = \mathbb{E}_{s_t \sim D}\left[\mathbb{E}_{a_t \sim \pi_\psi}(\beta \log(\pi_\psi(a_t|s_t)) - q_{\pi_\psi}(s_t, a_t))\right] \tag{5.11}$$

[1]Notice that, in Algorithm 1, D_{model} is cleared every time a context change is detected in order to avoid negative transfer from experiences drawn from previous contexts.

6 EXPERIMENTS

We evaluate MBCD in challenging continuous-state, continuous-action non-stationary environments, where the non-stationarity may result from qualitatively different reasons—ranging from abrupt changes to the system's dynamics (such as changes to the configuration of the agent's workspace); external latent environmental factors that impact the distribution over next states (such as random wind); robot malfunctions; and changes to the agent's goals (its reward function). We compare MBCD both against state-of-the-art RL algorithms and against state-of-the-art meta-learning methods specifically tailored to deal with non-stationary environments.

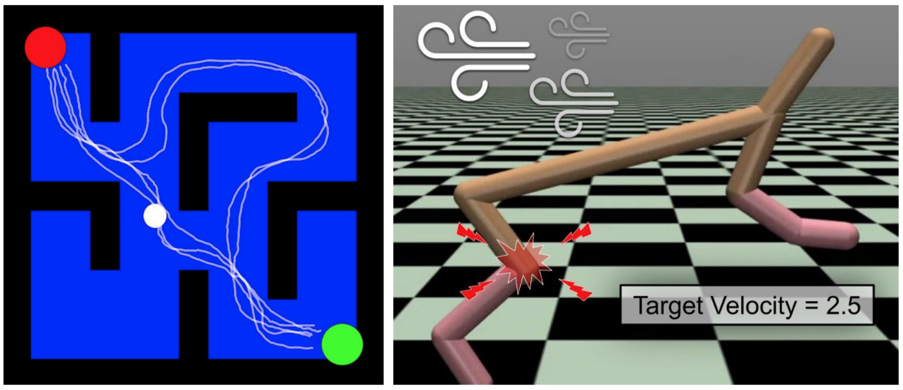
We investigate the performance of MBCD in two settings: *(i)* a setting that satisfies all standard requirements made by meta-learning algorithms; in particular, that all contexts in M_z are structurally similar, and that the agent is tested on a distribution of contexts that matches the training distribution; and *(ii)* a more general setting where such requirements are not be satisfied: contexts may differ arbitrarily, and future contexts experienced by the agent may be off-distribution with respect to those sampled during the training phase. We show that our method outperforms both standard state-of-the-art RL algorithms and also specialized meta-learning algorithms in both settings.

In the following figures depicting our experimental results, each curve shows the mean and shaded areas present information about the standard deviation, when running the experiments with different random seeds. We used 20 different random seeds in all experiments, except for the experiments in Fig. 6.2 and Fig. 6.5, in which we used 7 random seeds.

6.1 Environments

In what follows, we evaluate MBCD in two domains with qualitatively different non-stationary characteristics (see Fig. 6.1).

Figure 6.1: (a) Non-Stationary Continuous Particle Maze; (b) Half-Cheetah in a Non-Stationary World.



6.1.1 Non-Stationary Continuous Particle Maze

This domain simulates a family of two-dimensional continuous mazes where a particle must reach a non-observable goal location.

- Observation. $S_t = [x_{pos}, y_{pos}]$. The observed state consists of the particle coordinates in the continuous grid.
- Action. $A_t \in [-1.0, 1.0]^2$. Represents the direction of movement along the two dimensions.
- Reward. $R_t = -||S_t - goal||_2$. The reward function corresponds to the negative of the Euclidean distance between the particle and the goal location. The agent also receives a bonus reward of $+1$ when it is near the goal location.

Non-stationarity is introduced by either changing the location of walls or by randomly changing the latent goal location.

6.1.2 Half-Cheetah in a Non-Stationary World

This domain consists in a simulation of the high-dimensional *Half-Cheetah* robot from OpenAI Gym (BROCKMAN et al., 2016), using MuJoCo's physics engine (TODOROV; EREZ; TASSA, 2012). *Half-Cheetah* agent is made up of 7 rigid links (1 for torso, 3 for forelimb, and 3 for hindlimb), connected by 6 joints.

- Observation. $S_t \in \mathbb{R}^{17}$. The observed state is given by a 17-dimensional real-valued vector, including the position and velocity of the agent's center of mass and the angular position and angular velocity of each of its 6 joints.

- Action. $A_t \in [-1.0, 1.0]^6$. The action vector is the torque applied to each one of the agent's 6 joints.
- Reward. $R_t = -|v_x - v_g| - 0.1||A_t||^2$, where v_x is the agent's observed velocity along the x-axis, and v_g is the target velocity. The agent's goal is to move forward while reaching a target-velocity and minimizing control costs.

We introduce three sources of non-stationarity:

1. **random wind**: an external latent horizontal force, opposite to the agent's movement direction, is applied;
2. **joint malfunction**: either the torque applied to a joint of the robot has its polarity/sign changed; or a joint is randomly disabled;
3. **target velocity**: the target velocity of the robot is sampled from the interval 1.5 to 2.5, causing a non-stationary change to the agent's reward function.

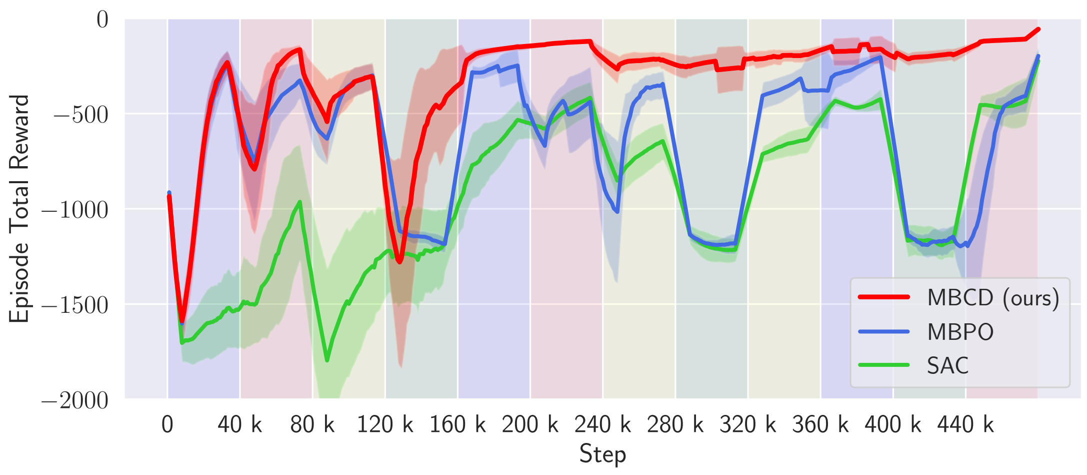
6.2 Results in the Non-Stationary Half-Cheetah Domain

We first evaluate our method on the non-stationary Half-Cheetah domain and compare it with two state-of-the-art RL algorithms: MBPO (JANNER et al., 2019) and SAC (HAARNOJA et al., 2018). In our setting, MBPO can be be seen as a particular case of our algorithm, where a single dynamics model and policy are tasked with optimizing behavior under changing contexts. SAC works similarly to MBPO but does not perform Dyna-style planning steps using a learned dynamics model.

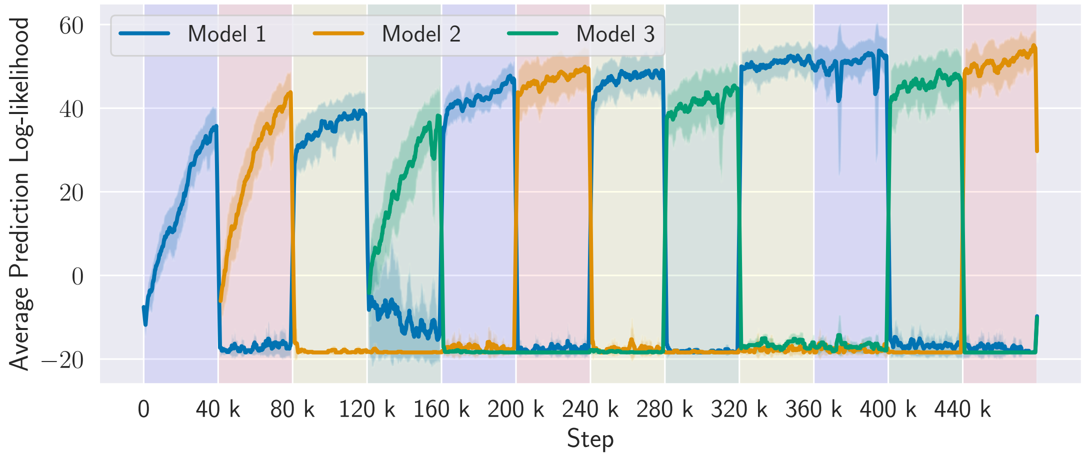
Fig. 6.2a shows the total reward achieved by different methods (ours, MBPO, SAC) as contexts change. Colored shaded areas depict different contexts, as discussed in the figure's caption. Notice that our method and MBPO have similar performances when interacting for the first time with the first three random contexts. In particular, both MBCD and MBPO's performances temporarily drop when a novel context is encountered for the first time. MBCD's performance drops because it instantiates a new dynamics model for the newly encountered context, while MBPO's performance drops because it undergoes negative transfer. SAC, which is model-free, never manages to achieve reasonable performance during the duration of each context, due to sample inefficiency. However, as the agent encounters contexts with structural similarities with respect to previously-encounters ones (around time step 160k), MBCD's performance becomes near-optimal:

it rapidly identifies whenever a context change has occurred and deploys an appropriate policy. Notice that MBCD modeled the wind context (yellow area) using the same model as the default context (blue area). This is because wind did not introduce a significant change to the MDPs state transition function. Consequently, MBCD automatically inferred that a single policy could perform well in both contexts and operated without significant reward loss in the long term—see, e.g., time steps 320k to 400k. MBPO and SAC, on the other hand, suffer from negative transfer due to learning average policies or dynamics models. They are also subject to catastrophic forgetting and do not reuse previously-acquired, context-specific knowledge.

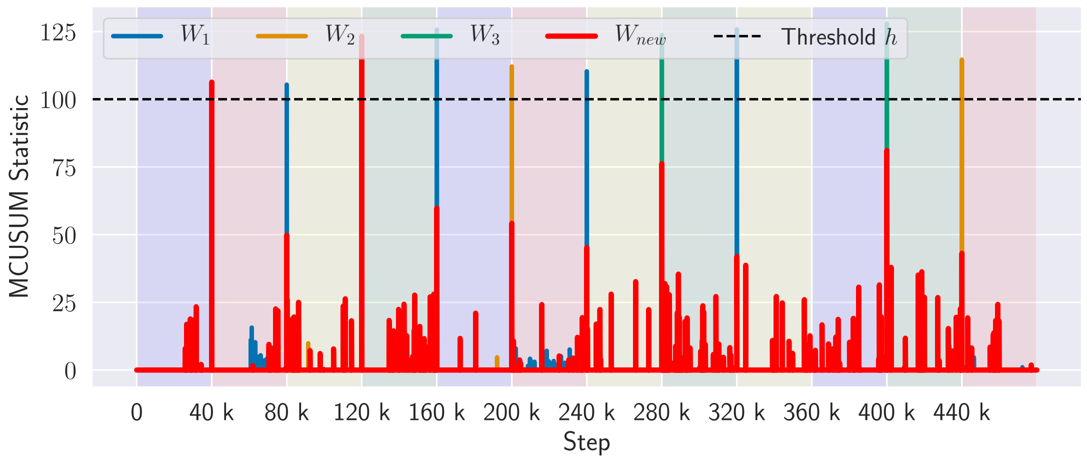
Figure 6.2: Evaluation of MBCD on the non-stationary *Half-Cheetah* domain. Colored shaded areas represent different contexts: *(blue)* default context; *(red)* joint malfunction; *(yellow)* wind; *(green)* novel target velocity.



(a) Total reward achieved by different methods (MBCD, MBPO, and SAC) as contexts change.



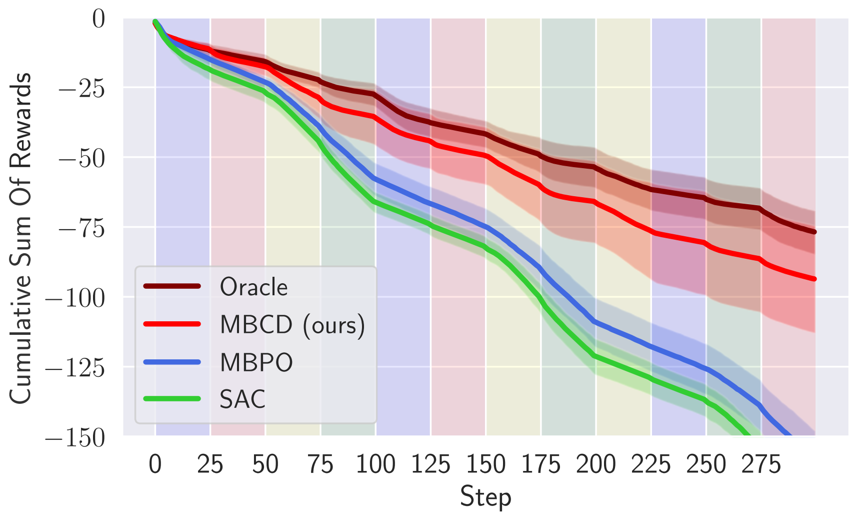
(b) Log-likelihood of the predictions made by each context model learned by MBCD as contexts change.



(c) Time evolution, as contexts change, of the MCUSUM statistics, W_k, for each model k. A context change is detected online whenever one of the statistics crosses the threshold h.

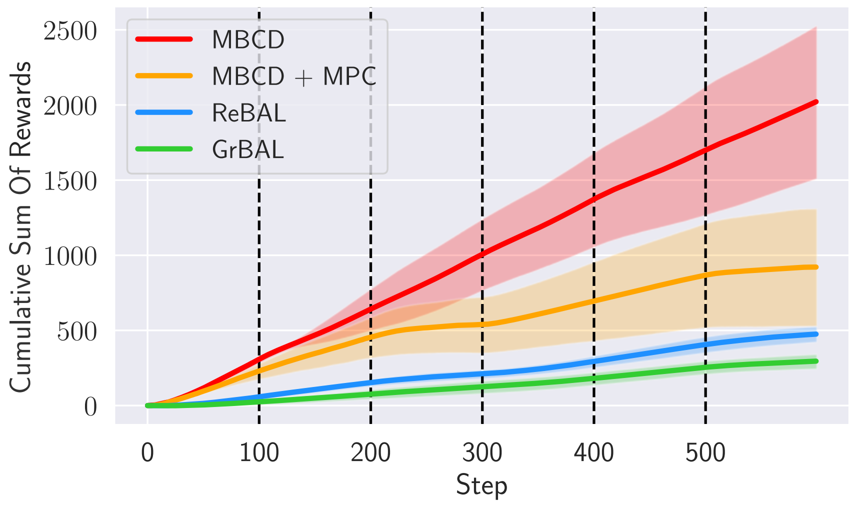
Fig. 6.2b and Fig. 6.2c allow us to observe the inner workings of MBCD and understand the reasons that underlie its performance. Fig. 6.2b shows the log-likelihood of the predictions made by each context model learned by MBCD as contexts change. Notice that all context changes in this experiment—even those caused by qualitatively different sources of non-stationarity—are detected with minimal delay. As contexts change, MBCD rapidly detects each change and instantiates new specialized joint prediction models for each context. Furthermore, when structurally similar contexts are re-encountered (e.g., at time steps 160k and timestep 360k), MBCD successfully recognizes that previously-learned models may be redeployed and avoids having to relearn context-specific dynamics or policies. Fig. 6.2c shows the time evolution, as contexts change, of the MCUSUM statistics, W_k, for each model k in the ensemble. A context change is detected, online, whenever one of the statistics crosses the threshold h. Notice that MCUSUM statistics grow rapidly and cross the threshold almost instantaneously—only a few steps after a context change. This empirically confirms the minimum-delay guarantees provided by our change-point detection method.

Figure 6.3: Regret of different methods (MBCD, MBPO, SAC) in the non-stationary *Half-Cheetah* domain, as contexts change rapidly, compared to an Oracle algorithm.



We now evaluate MBCD's ability to rapidly detect context changes after an initial training period. Fig. 6.3 shows the cumulative sum of rewards achieved by MBCD, MBPO, SAC, and by an Oracle algorithm that is initialized with optimal policies for all contexts and that detects context changes with zero delay. This is a challenging setting where contexts change very rapidly—after only 25 steps. Notice that our algorithm closely matches the performance of the zero-delay Oracle, thus empirically confirming its ability to minimize regret (Eq. 4.5).

Figure 6.4: Performance of MBCD and meta-learning methods (after a pre-training phase) in the *Half-Cheetah* domain with non-stationary malfunctions that disable random joints. Vertical dashed lines indicate context changes.



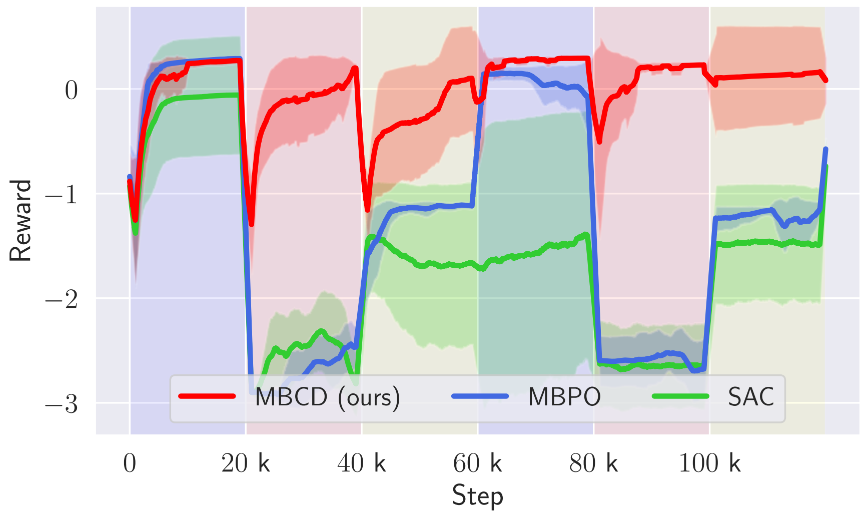
Next, we analyze how MBCD performs when compared with state-of-the-art meta-learning methods specifically tailored to deal with non-stationary environments: Gradient-Based Adaptive Learner[1] (GrBAL) (NAGABANDI et al., 2019) and Recurrence-Based Adaptive Learner[1] (ReBAL) (NAGABANDI et al., 2019). Both methods were detailed previously in Section 3.3.

Fig. 6.4 compares the adaptation performance of MBCD and the meta-learning methods in a non-stationary setting where (inspired by (NAGABANDI et al., 2019)) random joints of the Half-Cheetah robot are disabled after every 100 time steps. In this experiment we compare MBCD, ReBAL, GrBAL, and also (for fairness) a variant of MBCD that chooses actions using MPC instead of SAC. All implementations of MPC make use of the cross entropy method (CEM) (BOTEV et al., 2013) to accelerate action selection. All methods are allowed to interact with each randomly-sampled context during a training phase comprising 60000 time steps. Although the meta-learning methods have lower-variance, their meta-prior models do not perform as well as the MBCD context-specific dynamics models and policies. We also observe that when MBCD uses parameterized policies, learned through Dyna-style planning, it performs better than MBCD coupled with MPC. This confirms the findings of (JANNER et al., 2019) regarding the possible advantages of learning a parameterized policy instead of using MPC for action selection.

[1]We used the authors' implementation of the method, publicly available at <https://github.com/iclavera/learning_to_adapt>.

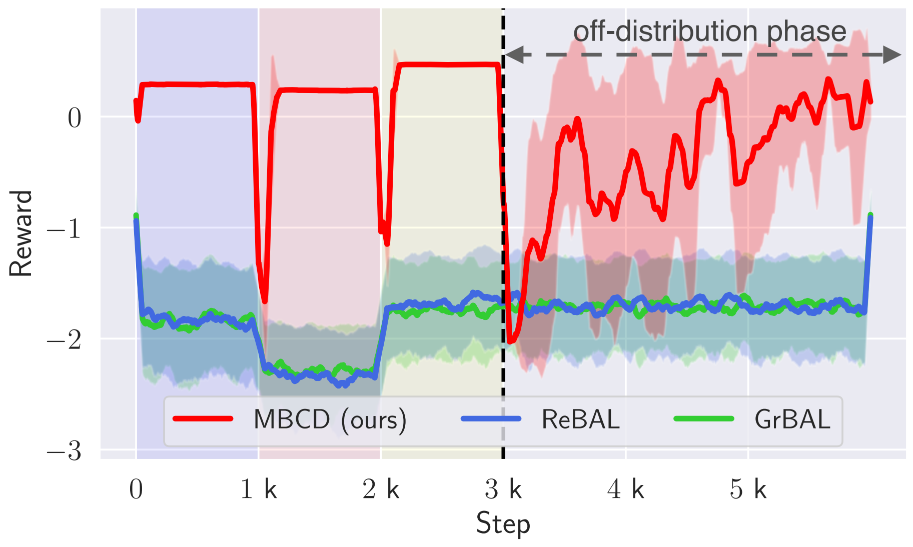
6.3 Results in the Continuous Particle Maze Domain

Figure 6.5: Rewards in the non-stationary continuous particle maze domain. Shaded colored areas indicate different contexts: *(blue)* default maze; *(red)* maze with non-stationary wall positions; *(yellow)* non-stationary target positions.



We now evaluate MBCD in a setting where contexts may differ arbitrarily and where future contexts may be off-distribution with respect to those sampled during the training phase. To do this, we compare MBCD, MBPO, SAC, ReBAL, and GrBAL, in the non-stationary continuous particle maze domain, where the sources of non-stationarity are as discussed earlier. Fig.6.5 compares MBCD, MBPO, and SAC in the fully-online setting—no pre-training phase is allowed. Notice that, even in this relatively simple scenario, state-of-the-art RL algorithms may fail if the underlying state transition or reward function changes drastically. MBCD's performance, by contrast, remains approximately constant (and high) as contexts change.

Figure 6.6: Rewards in the non-stationary maze domain. We introduce a phase with off-distribution contexts—contexts unlike those observed during pre-training.



In Fig. 6.6, we compare MBCD with meta-learning methods after a phase of pre-training. All contexts observed up to time step $3k$ are on-distribution: they are similar to those experienced during training. MBCD outperforms ReBAL and GrBAL because the meta-learning approaches construct models that try to average characteristics of structurally different contexts. At time $3k$, we initiate a phase with off-distribution contexts—contexts unlike those observed during training. When this occurs, MBCD faces a short adaption period and rapidly recovers, while meta-learning techniques perform poorly. This emphasizes MBCD's advantages over meta-learning models, both when a pre-training phase is not allowed/possible (e.g., Fig 6.5) and also when testing contexts arise from a distribution different from the training distribution.

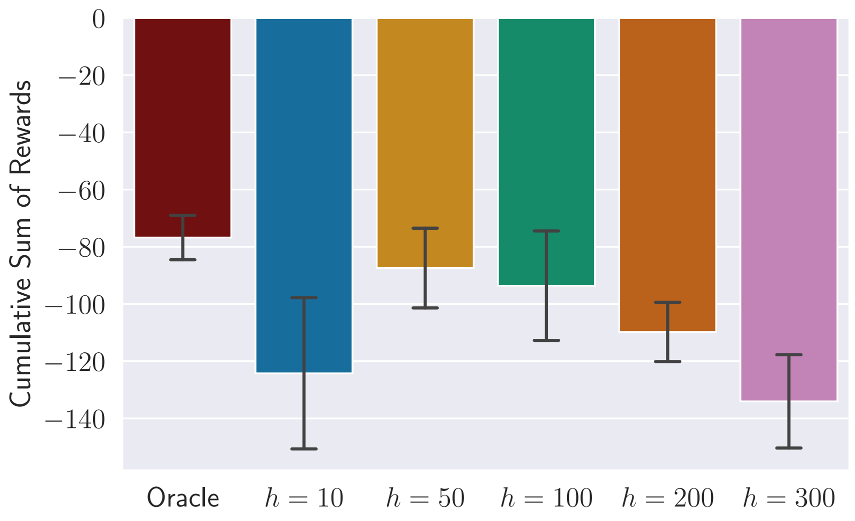
6.4 Sensitivity Analysis of the Detection Threshold Parameter

Next, we perform a sensitivity analysis on the impact of the value of the detection threshold parameter h on the MBCD performance. In Fig. 6.7, we show the impact of the value of the MCUSUM detection threshold h on the cumulative sum of rewards obtained with MBCD, and we compare it with the performance of an Oracle, which knows the change-points *a priori*. This experiment setting is the same as Fig 6.3, in which the *Half-Cheetah* undergoes a context change every 25 steps.

When the detection threshold is sufficiently small ($h = 10$), MBCD suffers from false alarms that mistakenly identify new contexts. For intermediate values ($h = 50$ and

$h = 100$), MBCD shows low regret and closely approximates the Oracle performance, as it does not present any false alarms. As the detection threshold increases ($h = 200$ and $h = 300$), the detection delay also increases, therefore resulting in high regret during the period in which the incorrect policy is being selected.

Figure 6.7: Impact of the MCUSUM detection threshold parameter h.



Notice that this experiment corresponds to an extreme case in which context changes happen very frequently, and the impact of the value of the detection threshold would be less significant in more regular scenarios.

7 CONCLUSION

We introduced a model-based reinforcement learning algorithm (MBCD) that learns efficiently in non-stationary settings with continuous states and actions. It makes use of high-confidence change-point detection statistics to detect context changes with minimum delay, while bounding the rate of false alarm. MBCD is capable of optimizing policies online, without requiring a pre-training phase, even when faced with streams of arbitrarily different contexts drawn from unknown distributions.

We empirically showed that it outperforms state-of-the-art (model-free and model-based) RL algorithms, and that it outperforms state-of-the-art meta-learning methods specially designed to deal with non-stationarity—in particular if the agent is faced with MDPs that are off-distribution with respect to the set of training contexts. We evaluated the algorithms in high-dimensional tasks where the non-stationarity result from qualitatively different reasons—ranging from abrupt changes to the system's dynamics (such as changes to the configuration of the agent's workspace); external latent environmental factors that impact the distribution over next states (such as random wind); robot malfunctions; and changes to the agent's goals (its reward function).

We also point out the advantage of model-based over model-free reinforcement learning algorithms in what regards fast adaptation. Our results emphasize the better sample-efficiency of model-based algorithms, which makes them capable of learning with orders of magnitude fewer environment interactions when a new context is introduced. Additionally, we observed that meta-learned models can not match the performance of specialized policies and, especially, that they struggle in scenarios in which testing contexts arise from a distribution different from the training distribution.

As future work, we would like to extend our method so that it can actively transfer policies between contexts. This research direction suggests that our method may be combined with meta-learning techniques that operate over policies, instead of over dynamics models. Another interesting direction is to explore multi-objective reinforcement learning problems (ROIJERS et al., 2013), in which multiple conflicting objectives must be considered. In this case, instead of different MDP dynamics or reward functions, the agent must quickly adapt to different user preferences regarding the objectives priorities.

REFERENCES

AMINIKHANGHAHI, S.; COOK, D. A survey of methods for time series change point detection. **Knowledge and Information Systems**, v. 51, set. 2016.

BANERJEE, T.; LIU, M.; HOW, J. P. Quickest change detection approach to optimal control in markov decision processes with model changes. **American Control Conference (ACC)**, IEEE, p. 399–405, 2017.

BASSO, E. W.; ENGEL, P. M. Reinforcement learning in non-stationary continuous time and space scenarios. In: **Proceedings of the 7th Brazilian Meeting on Artificial Intelligence (ENIA)**. [S.l.: s.n.], 2009. v. 7, p. 1–8.

BOTEV, Z. I. et al. Chapter 3 - the cross-entropy method for optimization. In: RAO, C.; GOVINDARAJU, V. (Ed.). **Handbook of Statistics**. [S.l.]: Elsevier, 2013, (Handbook of Statistics, v. 31). p. 35–59.

BROCKMAN, G. et al. Openai gym. **arXiv e-prints**, p. arXiv:1606.01540, jun. 2016.

CAMACHO, E.; ALBA, C. **Model Predictive Control**. [S.l.]: Springer London, 2013. (Advanced Textbooks in Control and Signal Processing). ISBN 9780857293985.

CHUA, K. et al. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: **Proceedings of the 32nd International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2018. (NIPS'18), p. 4759–4770.

DEISENROTH, M. P.; RASMUSSEN, C. E. Pilco: A model-based and data-efficient approach to policy search. In: **Proceedings of the 28th International Conference on International Conference on Machine Learning**. Madison, WI, USA: Omnipress, 2011. (ICML'11), p. 465–472. ISBN 9781450306195.

DUAN, Y. et al. Rl^2: Fast reinforcement learning via slow reinforcement learning. **arXiv e-prints**, p. arXiv:1611.02779, nov. 2016.

FINN, C.; ABBEEL, P.; LEVINE, S. Model-agnostic meta-learning for fast adaptation of deep networks. In: PRECUP, D.; TEH, Y. W. (Ed.). **Proceedings of the 34th International Conference on Machine Learning (ICML)**. International Convention Centre, Sydney, Australia: PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 1126–1135. Disponível em: <http://proceedings.mlr.press/v70/finn17a.html>.

GARCíA, C. E.; PRETT, D. M.; MORARI, M. Model predictive control: Theory and practice—a survey. **Automatica**, v. 25, n. 3, p. 335–348, 1989. ISSN 0005-1098.

HAARNOJA, T. et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: DY, J.; KRAUSE, A. (Ed.). **Proceedings of the 35th International Conference on Machine Learning**. Stockholmsmässan, Stockholm Sweden: PMLR, 2018. (Proceedings of Machine Learning Research, v. 80), p. 1861–1870. Disponível em: <http://proceedings.mlr.press/v80/haarnoja18b.html>.

HADOUX, E.; BEYNIER, A.; WENG, P. Sequential decision-making under non-stationary environments via sequential change-point detection. In: **Learning over Multiple Contexts (LMCE)**. Nancy, France: [s.n.], 2014.

HEALY, J. D. A note on multivariate cusum procedures. **Technometrics**, [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality], v. 29, n. 4, p. 409–412, 1987. ISSN 00401706. Disponível em: <http://www.jstor.org/stable/1269451>.

JANNER, M. et al. When to trust your model: Model-based policy optimization. In: WALLACH, H. et al. (Ed.). **Advances in Neural Information Processing Systems (NIPS) 32**. Curran Associates, Inc., 2019. p. 12519–12530. Disponível em: <http://papers.nips.cc/paper/9416-when-to-trust-your-model-model-based-policy-optimization.pdf>.

KAPLANIS, C.; SHANAHAN, M.; CLOPATH, C. Policy consolidation for continual reinforcement learning. In: CHAUDHURI, K.; SALAKHUTDINOV, R. (Ed.). **Proceedings of the 36th International Conference on Machine Learning (ICML)**. Long Beach, California, USA: PMLR, 2019. (Proceedings of Machine Learning Research, v. 97), p. 3242–3251.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: BENGIO, Y.; LECUN, Y. (Ed.). **3rd International Conference on Learning Representations (ICLR)**. San Diego, CA, USA: [s.n.], 2015.

LAI, T. L. Information bounds and quick detection of parameter changes in stochastic systems. **IEEE Transactions on Information Theory**, v. 44, n. 7, p. 2917–2929, 1998.

LAKE, B. M.; SALAKHUTDINOV, R.; TENENBAUM, J. B. Human-level concept learning through probabilistic program induction. **Science**, American Association for the Advancement of Science, v. 350, n. 6266, p. 1332–1338, 2015. ISSN 0036-8075. Disponível em: <https://science.sciencemag.org/content/350/6266/1332>.

LAKSHMINARAYANAN, B.; PRITZEL, A.; BLUNDELL, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In: **Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)**. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 6405—-6416. ISBN 9781510860964.

LAM, K.; YAM, H. Cusum techniques for technical trading in financial markets. **Asia-Pacific Financial Markets**, v. 4, p. 257–274, jan. 1997.

LORDEN, G. Procedures for reacting to a change in distribution. **The Annals of Mathematical Statistics**, v. 42, dez. 1971.

MAHMOUD, M. A.; MARAVELAKIS, P. E. The performance of multivariate cusum control charts with estimated parameters. **Journal of Statistical Computation and Simulation**, Taylor & Francis, v. 83, n. 4, p. 721–738, 2013. Disponível em: <https://doi.org/10.1080/00949655.2011.633910>.

MOERLAND, T. M.; BROEKENS, J.; JONKER, C. M. Model-based reinforcement learning: A survey. **arXiv e-prints**, p. arXiv:2006.16712, jun. 2020.

NAGABANDI, A. et al. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In: **7th International Conference on Learning Representations (ICLR)**. New Orleans, LA, USA: [s.n.], 2019.

NAGABANDI, A.; FINN, C.; LEVINE, S. Deep online learning via meta-learning: Continual adaptation for model-based rl. In: **7th International Conference on Learning Representations (ICLR)**. New Orleans, LA, USA: [s.n.], 2019. abs/1812.07671.

NAGABANDI, A. et al. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In: **2018 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2018. p. 7559–7566.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th International Conference on International Conference on Machine Learning**. Madison, WI, USA: Omnipress, 2010. (ICML'10), p. 807–814. ISBN 9781605589077.

PADAKANDLA, S. A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments. **arXiv e-prints**, p. arXiv:2005.10619, maio 2020.

PAGE, E. S. Continuous inspection schemes. **Biometrika**, [Oxford University Press, Biometrika Trust], v. 41, n. 1/2, p. 100–115, 1954. ISSN 00063444. Disponível em: <http://www.jstor.org/stable/2333009>.

POLLAK, M. Optimal detection of a change in distribution. **The Annals of Statistics**, v. 13, mar. 1985.

RAKELLY, K. et al. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In: CHAUDHURI, K.; SALAKHUTDINOV, R. (Ed.). **Proceedings of the 36th International Conference on Machine Learning (ICML)**. Long Beach, California, USA: PMLR, 2019. (Proceedings of Machine Learning Research, v. 97), p. 5331–5340.

ROIJERS, D. M. et al. A survey of multi-objective sequential decision-making. **Journal of Artificial Intelligence Research**, AI Access Foundation, El Segundo, CA, USA, v. 48, n. 1, p. 67–113, out. 2013. ISSN 1076-9757.

ROLNICK, D. et al. Experience replay for continual learning. In: WALLACH, H. et al. (Ed.). **Advances in Neural Information Processing Systems (NIPS) 32**. Curran Associates, Inc., 2019. p. 350–360. Disponível em: <http://papers.nips.cc/paper/8327-experience-replay-for-continual-learning.pdf>.

SAHKI, N.; GEGOUT-PETIT, A.; WANTZ-MÉZIÈRES, S. Performance study of change-point detection thresholds for cumulative sum statistic in a sequential context. **Quality and Reliability Engineering International**, Wiley, n. 1–21, p. 21, jul. 2020.

SEKAR, R. et al. Planning to explore via self-supervised world models. In: III, H. D.; SINGH, A. (Ed.). **Proceedings of the 37th International Conference on Machine Learning**. Virtual: PMLR, 2020. (Proceedings of Machine Learning Research, v. 119), p. 8583–8592.

SIBANDA, T.; SIBANDA, N. The cusum chart method as a tool for continuous monitoring of clinical outcomes using routinely collected data. **BMC medical research methodology**, v. 7, p. 46, nov. 2007.

SILVA, B. C. d. et al. Dealing with non-stationary environments using context detection. In: COHEN, W. W.; MOORE, A. (Ed.). **Proceedings of the 23rd International Conference on Machine Learning ICML**. New York, ACM Press, 2006. p. 217–224. Disponível em: <www.inf.ufrgs.br/maslab/pergamus/pubs/Silva+2006icml.pdf>.

SUTTON, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: **Proceedings of the 7th International Conference on Machine Learning**. [S.l.: s.n.], 1990. p. 216–224.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. Second. Cambridge, Massachusetts, USA: The MIT Press, 2018.

SUTTON, R. S. et al. Dyna-style planning with linear function approximation and prioritized sweeping. In: **Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence**. Arlington, Virginia, USA: AUAI Press, 2008. (UAI'08), p. 528–536. ISBN 0974903949.

TODOROV, E.; EREZ, T.; TASSA, Y. Mujoco: A physics engine for model-based control. In: **2012 IEEE/RSJ International Conference on Intelligent Robots and Systems**. Vilamoura, Portugal: [s.n.], 2012. p. 5026–5033.

VEERAVALLI, V.; BANERJEE, T. Quickest change detection. **Academic Press Library in Signal Processing**, v. 3, out. 2012.

VINYALS, O. et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. **Nature**, v. 575, p. 350—-354, nov. 2019.

YU, C.; LIU, J.; NEMATI, S. Reinforcement learning in healthcare: A survey. **arXiv e-prints**, p. arXiv:1908.08796, ago. 2019.

ZHOU, W. et al. Reinforcement learning experience reuse with policy residual representation. In: **Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)**. Macao, SAR, China: [s.n.], 2019.

ZIEMKE, T.; ALEGRE, L. N.; BAZZAN, A. L. A reinforcement learning approach with fourier basis linear function approximation for traffic signal control. In: **Proceedings of the Eleventh Workshop on Agents in Traffic and Transportation (ATT-2020)**. Virtual: CEUR-WS.org, 2020.

APPENDIX A — PARAMETER SETTINGS

In Table A.1 we show the parameters used for MBCD in the *Half-Cheetah* and the *Continuous Particle Maze* domains.

Table A.1: MBCD parameters.

Environment	Half-Cheetah	Continuous Particle Maze
N	5	5
Dynamics architecture	4 layers with 200 neurons	2 layers with 32 neurons
F	250	250
L	10^5	10^5
Policy architecture	2 layers with 256 neurons	2 layers with 64 neurons
h	100	1000
δ	2	2.5

The neural networks used to model both the dynamics and the policies are Multi-Layer Perceptrons (MLP) with Rectified Linear Units (ReLU) activation function (NAIR; HINTON, 2010). They were trained with mini-batch gradient descent using the Adam optimizer (KINGMA; BA, 2015).

The parameters used for SAC and MBPO were the same (when applied) parameters as in Table A.1.

For GrBAL and ReBAL, we used a reference implementation provided by the authors[1]. The parameters used for the *Half-Cheetah* domain were the same as in the original paper (NAGABANDI et al., 2019). A reduced number of layers and neurons were used for the *Continuous Particle Maze* domain. Regarding the action selection using MPC, we used the CEM method with 1000 candidate actions and planning horizon equal to 20 for all methods.

[1]https://github.com/iclavera/learning_to_adapt