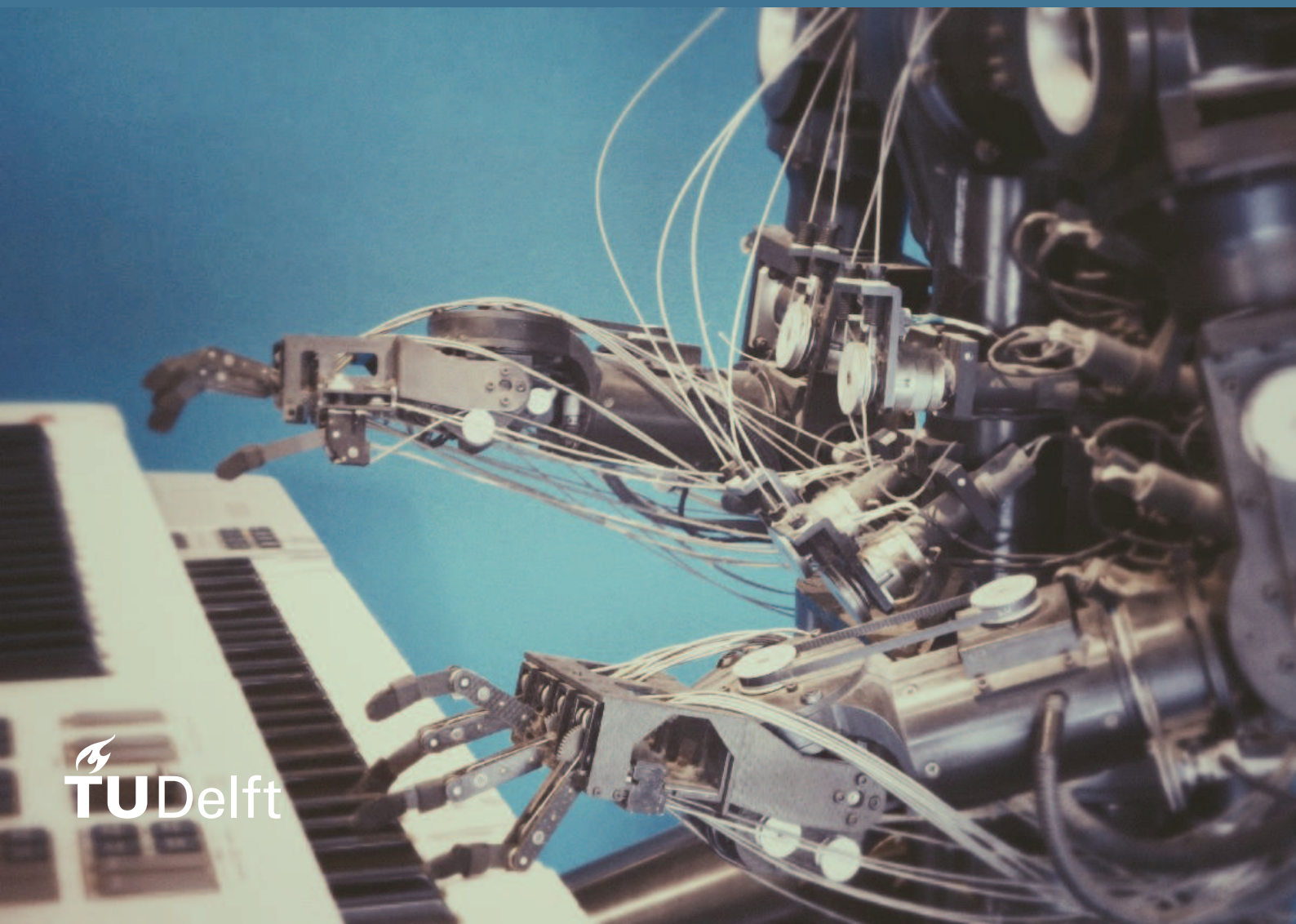
Policy Learning with Human Teachers

Using directive feedback in a Gaussian framework

D. Wout

Master Thesis
Systems & Control
March 27th, 2019



TU Delft

Policy Learning with Human Teachers

Using directive feedback in a Gaussian framework

by

D. Wout

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday March 27, 2019 at 10:30 AM.

Student number:	4247833	
Project duration:	February 15, 2018 - March 15, 2019	
Thesis committee:	prof. dr. ir. D. Gavrila	chair
	dr. ir. J. Kober	supervisor
	dr. C. Celemin Paez	supervisor
	dr. M. Kok	

Cover image is "robot playing piano" by Franck V. on Unsplash.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

A prevalent approach for learning a control policy in the model-free domain is by engaging Reinforcement Learning (RL). A well known disadvantage of RL is the necessity for extensive amounts of data for a suitable control policy. For systems that concern physical application, acquiring this vast amount of data might take an extraordinary amount of time. In contrast, humans have shown to be very efficient in detecting a suitable control policy for reference tracking problems (Chernova and Thomaz, 2014). Employing this intuitive knowledge has proven to render model-free learning strategies suitable for physical applications (Meriçli et al., 2011). Recent studies have shown that learning a policy by directive action corrections is a very efficient approach in employing this domain knowledge. Moreover, feedback based methods do not necessarily require expert knowledge on modelling and control and are therefore more generally applicable. The current state-of-the-art regarding directional feedback was introduced by Celemin and Ruiz-del Solar (2015) and coined COrrective Advice Communicated by Humans (COACH). In this framework the trainer is able to correct the observed actions by providing directive advise for iterative policy updates. However, COACH employs Radial Basis Function (RBF) networks, which limit the capabilities to apply the framework on higher dimensional problems due to an infeasible tuning process.

This study introduces Gaussian Process Coach (GPC), an algorithm preserving COACH's structure, but introducing Gaussian Processes (GPs) as alternative to RBF networks. Moreover, the employment of GPs allows for uncertainty estimation of the policy, which will be used for 1) inquiring high-informative feedback samples in an Active Learning (AL) framework, 2) introduce an Adaptive Learning Rate (ALR) that adapts the learning rate to the coarse or refine focused learning phase of the trainer and 3) establish a novel sparsification technique that is specifically designed for iterative GP policy updates. We will show by employing synthesized and human teachers that the novel algorithm outperforms COACH on every domain tested, with the most outspoken difference on higher dimensional problems. Furthermore, we will prove the independent contributions of AL and ALR.

“ A library is the delivery room for the birth of ideas,
a place where history comes to life. ”

— *Norman Cousins*

Contents

1 Introduction

As robots start pervading our daily environment, the need for adaptive and tailored behavior is becoming greater. Imagine an autonomous car that is fully adapted to the user's way of driving, or an autonomous vacuum cleaner that optimizes its trajectory based on the mapping of a house. The challenge here is that the control software should be designed according the requirements for the specific application or user, which may furthermore evolve over time. Model-based control design is then intractable, since no general model can be derived that accounts for all potential applications. Instead, such problems require model-free techniques. A prevalent control implementation in this field is Reinforcement Learning (RL), an approach that maximizes a measure of performance by trial and error. The measure of performance is usually designed as a function of sensory values, where the objectives are rewarded and unwanted behaviour is possibly penalized. Finding a suitable reward function, especially one that continually respects the purposes of the user, is however a challenge and a subject of research in its own right. Moreover, the biggest challenge in RL that hinders widespread application is that decent control requires a vast amount of input-output data (Kober et al., 2013). This data is easy to acquire for simulated environments (e.g. computer games), but rather problematic for real physical applications, such as the examples given earlier.

In contrast to RL, human tend to be very efficient in inferring a suitable control law when facing new problems. A study by Hessel et al. (2018) described how human participants took minutes to master an Atari game, whereas a RL implementation needed several days. Past studies have shown that by employing the human domain knowledge into the learning process, an optimal solution for control can be found much more efficiently. Let us consider Fig. 1.1, where it is shown that a human can actively participate in the learning process by invoking interaction with the agent. Interaction-based algorithms have shown to outperform autonomous learning techniques significantly on intuitive problems (e.g. Argall et al., 2009). A well known approach is the Learning from Demonstration (LfD) framework where a policy is derived using examples of proper execution. Other methods, like Apprenticeship Learning, employ demonstrations to reversely derive the trainer's choices for autonomous improvement (Abbeel and Ng, 2004). To avoid superfluous (and possibly expensive) interaction with the trainer, the LfD framework can be extended with Active Learning (AL), where demonstration are inquired especially for uncertain policy executions (Gräve and Behnke, 2013). LfD could however be troublesome for systems that feature agile dynamics. Moreover, the demonstrations require expert knowledge of the system and the solution and are hence not suitable for everyone to execute.

A second, less demanding and cognitive approach concerns policy learning by *evaluative feedback*, as studied by Knox and Stone (2009). In this framework, the trainer provides scalar values that qualify the actions of the agent. Although this implementation does not require expert knowledge, Thomaz et al. (2006) argue that this does not fully utilize people's intuition. Furthermore, Suay and

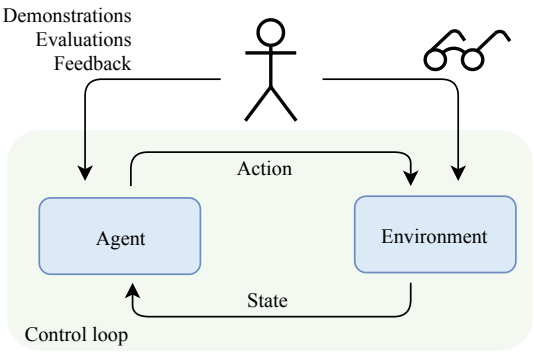


Figure 1.1: Schematic of how humans fit in the control loop. The 'Environment' (e.g. robot) is observed by the human who provides demonstrations/evaluations/feedback, depending on the applied algorithm. The embedded control policy in the 'Agent' is subsequently updated.

Chernova (2011) motivate that evaluative approaches do not scale well to more complex problems, in contrast to algorithms that employ corrective advice. This third means of human-machine interaction concerns the guidance of the agent by means of *directive feedback*. In comparison to evaluative feedback, corrections are more information-rich, whilst it is still not required to have expert knowledge on the task or control. Argall et al. (2008) introduced a pioneering approach to learning from corrective feedback that used *advice operators* (specific, static corrections). However, broad application was problematic since these operators had to be programmed a priori, again requiring the necessary knowledge. A more general and applicable approach was studied by Celemin and Ruiz-del Solar (2015) with the introduction of the current state-of-the-art, COrrective Advice Communicated by Humans (COACH). COACH concerns a framework that learns a continuous policy by corrective advise of trainers in the action-space. The methods outperform evaluative-based implementations by complying with the intrinsic preference of human to guide. However, COACH entails poor scaling abilities to higher order systems due to the employment of Radial Basis Function (RBF) networks, which concern an extensive design procedure. The potential of the feedback principle is thereby partly undone by the meticulous tuning process.

This study is hence focused on the extension of COACH that improves on the points of scaling. We introduce Gaussian Process Coach (GPC), a directive feedback implementation that introduces Gaussian Processes (GPs) as an alternative to RBF networks. We will show that GPC has better scaling abilities to higher dimensional problems compared to COACH. Moreover, by utilizing the uncertainty estimates of the GPs we will establish advantages by 1) inquiring high-informative feedback samples in an AL framework, 2) adapt the learning rate to the learning phase of the teacher and 3) introduce a novel sparsification technique for iterative GP updates.

This report is organized as follows: in Chapter 2 the essential background for this study is covered. Chapter 3 will detail the novel algorithm, GPC. This algorithm is benchmarked using the experimental setup described in Chapter 4. The results will be presented and discussed in Chapter 5. Finally, the study will be concluded with a conclusions and potential future research directions in Chapter 6.

2

Background and Related Work

This chapter will provide an overview of the relevant concepts and the associated literature. The items are assumed essential for good understanding of this study.

Section 2.1 will present the concept of model-free autonomous learning: Reinforcement Learning (RL). RL concerns a technique where a decision making policy is derived by trial and error. The mathematical background and the concepts most relevant to this study will be introduced.

Autonomous learning can be assisted by humans due to their intuitive domain knowledge. This information can be communicated by either evaluations, feedback samples or by demonstrating the complete task. All the methods have their advantages and drawbacks, which will be covered extensively in Section 2.2. Section 2.3 will introduce the current state-of-the-art of directional feedback implementation. This framework was studied by Celemin and Ruiz-del Solar (2015) and called COrrective Advice Communicated by Humans (COACH). However, COACH employs Radial Basis Function (RBF) networks as function approximator, which have some major drawbacks in terms of scaling. We therefore introduce Gaussian Processes (GPs) as an alternative in Section 2.4. The chapter will conclude with a conclusion in Section 2.5

2.1. Reinforcement Learning (RL)

The field of Reinforcement Learning (RL) concerns autonomous learning from interaction with the environment. The nature of learning becomes clear when we observe infants and humans in general: they interact with the environment and observe when the executed actions yield the consequences they intend. The action itself and the direct output of this action is less of concern than its effect: think of throwing a ball in a basket: the distance of the ball to the basket is more important than the angle of the arm, for instance.

In RL the same concept is employed, but than in an environment with a *computational agent*. Here, the output is not directly clear, but the effect is expressed in a *reward*-function. The learner must, as a matter of fact, discover which actions obtain the highest return by means of trial and error and develop a suitable sequential decision making policy. In this process no human knowledge is employed. The learned behavior is entirely dependent on the explored actions.

RL can neither be regarded as *supervised* or *unsupervised* learning (see Sutton and Barto, 2018). The performance of an algorithm is solely dependent on the obtained reward, which is determined by its current state and executed action. A converged algorithm can be considered as a system that knows the path towards a goal by yielding the maximum amount of accumulated reward.

Section 2.1.1 will introduce the framework that is used for modeling decision making policies, called Markov Decision Process (MDP). The policy and concept relevant to this study will be introduced here. Section 2.1.2 will dive into the applicability of RL to real-world systems.

2.1.1. Markov Decision Process (MDP)

A Markov Decision Process (MDP) provides a mathematical framework for modeling decision making situations. In a MDP-framework, actions do not influence only immediate reward, but also consecutive situations and states, and therefore future rewards. The frameworks provides a theoretical idealization that allows for predictions about future states and rewards.

The mathematical formalization of an MDP is described by a tuple $\langle S, A, T, R, \gamma \rangle$, where S denotes a set of states and A a set of actions, which can both be continuous or discrete. The transition function is denoted by T, which is defined as $T : S \times A \times S \rightarrow [0,1]$, and represents the probability of the respective state-transition. The reward per state-action pair is contained in $R : S \times A \times S \rightarrow \mathbb{R}$. The discount factor is defined as $\gamma \in [0,1)$ and represents scaling (importance) of future rewards. When the state s_{t+1} is solely determined by s_t and action a_t, we say that the system is *memoryless* and therefore satisfies the *Markov property*.

Important elements in RL framework concern the *policy*, that provides a mapping from state to action, and the *state(-action) value function*, which contains a measure of performance that the policy is supposed to maximize. Both features will be covered more extensively next.

Policy

A policy is a rule, that given the state $s_k \in S$, produces an action $a_k \in A$. More formally, it concerns a function $\pi : S \rightarrow A$, that maps the state to an action. This mapping describes a *deterministic policy*. Alternatively, we introduce a stochastic policy as $\pi : S \times A \rightarrow [0,1]$, which is denoted as $\pi(a|s)$ s.t.

$$\sum_{a \in A} \pi(a|s) = 1.$$

Stochastic policies list their actions with the associated chance of executing them. The goal in RL is to learn a policy that maximizes the discounted sum of future rewards (i.e. state value function, given by (2.1)). The optimization regarding this policy search is enclosed in a trade-of between *exploration* and *exploitation*. Exploration denotes the deviation from the known path, trying to find a better alternative path that leads to the desired reference. Exploitation describes the execution of the known policy. A converged policy solely exploits, and is denoted by

$$\pi^*(s) = \arg\max_a Q^*(s,a),$$

with π^* the optimal policy and Q^* representing a converged value function, as will be detailed next.

State(-Action) Value function

Almost all RL algorithms involve estimating a value function, a measure for how good it is to be in a certain state. This measure if formally defined as the reward that is expected for reaching the reference, starting from the current state (Sutton and Barto, 2018). We define the *state-value function* that denotes the expected return for starting in s and following π thereafter. In the context of an MDP, we can define

$$V^\pi(s) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s \right\}, \tag{2.1}$$

Here, $\mathbb{E}$ is the expected value of a random variable given that the agent will follow π. Similarly, we can define the *state-action value function* (Watkins, 1989). This function describes the expected return in state s, taking action a and following π thereafter, i.e.

$$Q^\pi(s,a) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a \right\}.$$

The given value function are estimated during the learning process. The policy combines exploration and exploitation in a tuned settlement to find alternative better paths that lead to the final (desired) state. A more profitable path is defined as a trajectory that yields are higher accumulated return. This continuous search process is defined as RL.

2.1.2. Applicability to physical processes

RL techniques are data driven, which means that performance is achieved by acquiring vasts amounts of input-output data (Kober and Peters, 2012). Depending on the size of the problem this process can take an extraordinary amount of time (Heess et al., 2017). An example given by Sutton and Barto (2018) showed that mastering a TD-backgammon game by state-of-the-art RL techniques needed 300.000 simulated games for achieving human performance. Analogously, Hessel et al. (2018) showed that even for a simple Atari game days of experience were required for achieving any performance. For small and confined problems RL techniques have shown to be applicable for real-world applications, e.g., the study by Kober and Peters (2009), where robot motion primitives were learned by RL implementations. However, the applications are mostly limited to very specific problems consisting of a low dimensional state-space (Kober et al., 2013). Moreover, RL implementation face another problem, namely the requirement of a reward function. Control synthesis is fully guided by maximizing the return (accumulated reward). However, as motivated by Abbeel and Ng (2004), is determining an explicit reward function sometimes intractable. The study mentioned the example of driving a car on the highway. Since many desiderata, such as pedestrians, safe driving distance, maintaining a reasonable speed etc. are hard to quantify, it is very complex to model an applicable and complete reward function.

In contrast, humans have shown to be perfectly capable of inferring a suitable control strategy for such problems. In these cases, a 'reward function' is instantly constructed by human intuition. Moreover, in contrast to the data driven approach of RL, humans are able to achieve decent performance at first try when facing new problems. The study by Hessel et al. (2018) showed for example that humans took only minutes to master the Atari game that took days for RL techniques. Humans employ a very valuable cognitive feature, called intuition, that lacks in autonomous learning applications. A combination of both world, where RL benefits from human domain knowledge, would comprise a very fruitful cooperation. This combination will be detailed in the next section.

2.2. Learning with human interaction

This section will dig deeper into the employment of humans in the learning loop. Due to the enhanced learning capabilities, this feature has been studied quite some in the past. The associated literature can roughly be divided into two sections of different cognitive requirements. The first section concerns a low cognitive implementation which is called learning from *evaluative feedback*. In this framework, trainers provide a signal that qualifies the observed behavior of the agent in order to encourage contributing and discourage counterproductive actions. This approach is detailed in Section 2.2.1. The second method concerns a more demanding framework, called *feedback in the action domain*. In this framework a trainer guides the agent by directive advise or by examples of a proper executions of the designated task, which is explained in Section 2.2.2. To prevent superfluous and expensive interaction with the trainer, existing implementations can be extended with Active Learning (AL), where high-informative interaction is prioritized. This implementation will hence will covered in Section 2.2.3.

2.2.1. Evaluative feedback

The least cognitive method that is covered in this study is control synthesis by means of evaluative signals. The trainer observes the environment and the policy of the agent and reward good signals for contributing actions and punishment signals for counterproductive behavior.

A well-known study of evaluative feedback for control synthesis is the work by Knox and Stone (2009), who designed a framework which could be applied to domains without an explicit reward function. The policy was fully derived from the feedback signals from the human. The main outline was as follows: the human observes the actions and provides scalar signals (e.g. between zero and ten) to perform updates on the policy. This simple principle showed to be very effective on

confined problems and showed to outperform autonomous algorithm significantly. This study also lead to many more extensions to it. For example Vien and Ertel (2012), that extended the framework to continuous action spaces, or Vollmer and Hemion (2017) that replaced the scalar signals by a five-star rating framework. Analogously, Loftin et al. (2014) showed improvement by responding to the punishment- or reward-focused mindset of people. However, the study by Griffith et al. (2013) argued that humans are unable to communicate how good or how bad actions are in a way that is straightforward to interpret for computational agents. As an alternative, the work introduced a Bayesian optimized approach in modelling feedback relative to the current policy, taking uncertainty into account to compensate for inconsistent feedback.

Overall, the main advantage of employing evaluative feedback is the low cognitive demands for the trainer. It entails a very intuitive way of interaction with the agent. It is furthermore easy to implement, does not require a reward function, and little to no expert knowledge is required. Its downside however, according to Suay and Chernova (2011) are the poor scaling abilities to higher order systems. Moreover, according to Thomaz et al. (2006), humans tend to intrinsically guide future actions in their evaluative signals rather than to evaluate the past actions. The next sections will introduce implementations that have improved on these points.

2.2.2. Feedback in the action domain

This section will be concerned with feedback in the action domain. We will divide this feedback approach into two sections, based on the expert knowledge required. The first section concerns the Learning from Demonstration (LFD) domain, and corrective demonstration in particular. This implementation requires the most expert knowledge from the trainer regarding control and execution of the task. The second framework concerns *directive feedback*. An approach which requires less expertise on the task, but rather a notion of execution. Both methods will be covered here.

Learning and Corrections from Demonstrations

In this framework the policy is derived from proper executions from the trainer (Argall et al., 2009) in the LFD domain. The executions are considered as ground-truth and are generalized over the state-space. An alternative method of employing demonstrations concerns the Apprenticeship Learning framework, where the reward function or a systems model is reversely derived for autonomous improvement (Ng et al., 2000; Abbeel and Ng, 2004). For sub-optimal behavior, the trainer can provide additional (or exclude counterproductive) demonstrations. To avoid superfluous (and possibly expensive) interaction with the trainer, Gräve and Behnke (2013); Losey and O'Malley (2018) improved sample efficiency with an Active Learning (AL) framework where demonstrations are inquired especially for uncertain policy executions.

The framework of *Corrections from Demonstrations* does not concern a method that can be applied generally. First of all, the trainer needs to have a certain expert knowledge about the execution of the task. Secondly, the system might feature agile dynamics, which may hinder the proper execution of the designated task. A less cognitive and active interaction implementation is covered next in *Directional feedback*.

Directional feedback

A second means of feedback in the action domain concerns *Directional feedback*. A pioneering approach was conducted by Argall et al. (2008); Argall (2009), which introduced a framework, coined A-OPI, where advise operators could be communicated to the agent. Later work by Meriçli et al. (2011) applied this algorithm successfully on Bipad Walk systems. However, the application of advise operator has a disadvantage when it comes to generality, since the feedback was application specific and had to be programmed in advance, with the required programming knowledge.

Later work improved on this point by introducing a more general framework, e.g., Chernova and Thomaz (2014). This study introduced an algorithm that enabled the application of directive feed-

back implementations to the discrete action-space domain. The extension to continuous state and action spaces has been studied by Celemin and Ruiz-del Solar (2015), which the introduction of the current state-of-the-art regarding directional learning, COACH. The main outline of COACH is as follows. Analogously to evaluative implementations does the trainer observe the environment and the policy of the agent. When the trainer want to correct the observed action, he or she provides an increase or decrease signal in the respective action channel. The policy is subsequently updated into the suggested direction. The method of directive feedback has a lot of advantages. First, no explicit reward function is required for control synthesis, since the control law is fully established by the trainer. Moreover, no expertise on the task and modelling is required, only a notion of the execution suffices. The principle can basically be employed on any problem. Furthermore, according to Suay and Chernova (2011), the principle scales better to higher order problems compared to evaluative implementations. However, although the principle is very well promising, the framework itself hinders the widespread application due to extensive design procedures, as will be further elaborated in Section 2.3.

2.2.3. Active Learning (AL)

The studies by Gräve and Behnke (2013); Losey and O'Malley (2018) have shown that superfluous (and possibly expensive) interaction with the trainer can be prevented by encouraging feedback for uncertain executions. This extension to learning frameworks is coined Active Learning (AL).

The AL formalism is a rather comprehensive term according to Chernova and Thomaz (2014). It describes the extension of any human assisted learning algorithm to a bi-directional feedback framework. This means that instead of only human induced interaction, the agent communicated when this interactions would be most useful. The inference whether interaction is requested is usually done by a Bayesian framework.

The current literature in this field is mostly focused on the LfD application. A typical example is the study by Gräve and Behnke (2013), where demonstrations were inquired for uncertain executions of the Gaussian Process policy. In addition, Maeda et al. (2017); Losey and O'Malley (2018) also showed that by requesting demonstrations for uncertainty policy regions, a much higher sample efficiency could be established. A study that dissociates from the mainstream is the paper by Li et al. (2016), who employed AL to an *evaluative* feedback framework. The uncertainty was estimated by distance measures between the current state and the closest state (Euclidean resp.). Unfortunately it did not yield the intended action, presumable because a feedback sample per state does not trigger the optimal action instantaneously. Interestingly, there was no literature found that combines AL with a directive feedback implementation.

2.3. COACH

Due to the required expert knowledge in the LfD domain and the implicit guidance and poor scaling capabilities of evaluative implementations, as explained in Section 2.2, this study will be focused on directive feedback algorithms. This section will introduce the current state-of-the-art regarding directional feedback in continuous action spaces. The most recent and leading algorithm was developed by Celemin and Ruiz-del Solar (2015) and called COrrective Advice Communicated by Humans (COACH). This section will provide an overview of the properties and will assess its advantages and disadvantages. The main outline of COACH is presented in Algorithm 1 and will be used for future reference.

In Section 2.3.1 the main architecture of COACH is explained with reference to Fig. 2.1. The feature space will be explained more extensively in Section 2.3.2.

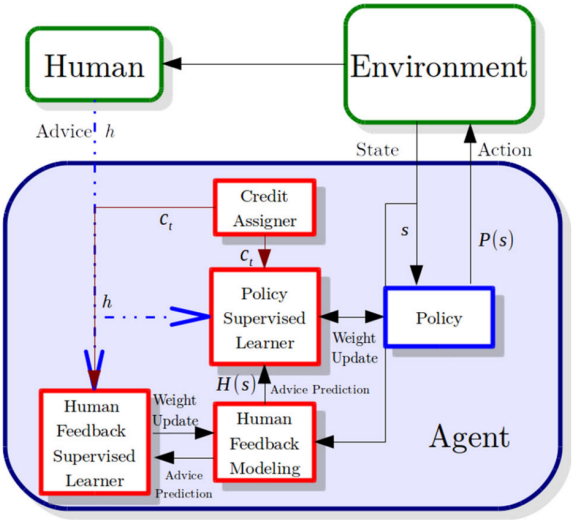


Figure 2.1: Architecture of COACH by Celemin and Ruiz-del Solar (2015). The policy is shaped by human corrective feedback only, no RL-algorithm is employed. Reprinted with permission of Celemin and Ruiz-del Solar.

2.3.1. Framework

COrrective Advice Communicated by Humans (COACH) is an algorithm for training agents by corrective advise. The formalism consists of a policy $P_c : S \rightarrow \mathbb{R}^n$, with S the set of states and n the action-space, that maps states to continuous actions. The human trainer observes the behavior of the agent and suggest occasionally to either increase or decrease the respective action. This feedback $h \in \{-1, 1\}$ is modelled in the *human feedback model*: $H_c : S \rightarrow \mathbb{R}^n$. The parametrisation of both models is done by Radial Basis Function (RBF) networks, where the respective models have different weight to the feature vector $\phi(\mathbf{s}_k)$, with $\mathbf{s}_k$ denoting the state in time-step k. The learning framework is supported by the following modules.

Policy Supervised Learner

This module models the policy $P_c(\mathbf{s}_k)$ of COACH. The policy provides the continuous action a_k to a given state $\mathbf{s}_k$, by taking the linear combination of the weights and the feature vector, i.e.

$$a_k = P_c(\mathbf{s}_k) = \theta_k^T \phi(\mathbf{s}_k),$$

with θ the weight vector of the policy. For every directive correction h given by the teacher, the weight vector is updated following a Stochastic Gradient Descent approach:

$$\theta_{k+1} = \theta_k - \alpha(\mathbf{s}_k) \nabla_\theta J(\theta),$$

with $\alpha(\mathbf{s}_k)$ the learning rate (obtained as described the next section) and $J(\theta)$ denoting the cost function, which is the squared error between the applied and 'desired' action, given by h and magnitude e. The latter denotes a free parameter set by the user within the range of the action domain. As a result, implementing the gradient in the the weight vector, yields

$$\theta_{k+1} = \theta_k + \alpha(\mathbf{s}_k) \phi(\mathbf{s}_k) he, \tag{2.2}$$

with e a parameter based on the correction (error) magnitude of the trainer.

Human Feedback Supervised Learner

This module models the feedback of the trainer as a function of the state $\mathbf{s}_k$. The predictions are carried out by a linear combination of the human model weights ψ_k and the feature vector $\phi(\mathbf{s}_k)$.

Algorithm 1 COACH framework

1: **Given:**
   Policy learning rate e
   Human model learning rate β
   Constant learning rate c_c
   Feature space function $\phi(\cdot)$
2: **for** all k **do**
3: Get state $\mathbf{s_k}$
4: Compute new action $a_k \leftarrow \theta_k \phi(\mathbf{s_k})$
5: Obtain corrective advise h
6: **if** $h \neq 0$ **then**
7: $H(\mathbf{s_k}) = \psi_k^T \phi(\mathbf{s_k})$
8: $\Delta\psi = \beta(h - H(\mathbf{s}_k))\phi(\mathbf{s}_k)$
9: Human model update $\psi_{k+1} = \psi_k + \Delta\psi$
10: Get learning rate $\alpha(\mathbf{s}_k) = |H(\mathbf{s}_k)| + c_c$
11: $\Delta\theta = \alpha(\mathbf{s}_k)\phi(\mathbf{s}_k)he$
12: Policy update $\theta_{k+1} = \theta_k + \Delta\theta$
13: **end if**
14: **end for**

The updates on the weight vector ψ_k are conducted in the same fashion as (2.2), but now with a known error magnitude $e_h = h - H(\mathbf{s}_k)$:

$$\psi_{k+1} = \psi_k + \beta(h - H(\mathbf{s}_k))\phi(\mathbf{s}_k),$$

with β the learning rate of the human model. From this human model, the learning rate in (2.2) is given by

$$\alpha(\mathbf{s}_k) = |H(\mathbf{s}_k)| + c_c, \tag{2.3}$$

Note that $H(\mathbf{s}_k) \approx 1$ for repeating feedback in the same direction, and therefore accelerates the learning process. For alternating feedback the learning rate diminishes. To prevent the learning rate for stagnating, (2.3) is appended with a constant factor c_c.

The outline of the COACH framework is depicted in Algorithm 1. Lines 3-5 comprise of policy executions. The update lines consist of the human prediction and human model updates (line 7-10). The policy updates are subsequently given in lines 11-12. Furthermore, the COACH framework can be extended by the *Credit Assigner*, which takes a human delay into account for the feedback given by the teacher. Since this study will not exploit this feature it will not be covered here.

2.3.2. Modelling the feature space

The human model H and the policy P are modelled by means of feature spaces, which concern a parameterized supervised learning method. Feature space modelling consist of two mappings: first a (usually) non-linear mapping from the input data into a feature space by basis functions. The second mapping concern a linear combinations of these basis functions. The combination of having both linearity and non-linearity makes them suitable and flexible for multiple purposes (Shin and Goel, 2000).

The feature space in COACH is given by a RBF network

$$P_c(\mathbf{s_k}) = \theta^{\mathbf{T}} \phi(\mathbf{s_k}), \quad \phi_i(\mathbf{s_k}) = \exp(-\frac{1}{2}(\mathbf{s}_k - \mu_i)^T \delta_i^{-1}(\mathbf{s}_k - \mu_i)), \tag{2.4}$$

where $\mathbf{s_k}$ denotes the state at time k and μ_i the center of the i'th basis function. The δ represents a tuning parameter. An alternative form of (2.4) is given by

$$P_c(\mathbf{s_k}) = \sum_{i=1}^{N} \theta_i \phi_i(\mathbf{s}_k), \tag{2.5}$$

where loop iterates over all N centers μ_i. The amount of centers is determined based on the roughness of the data. The more fluctuations present per input dimension, the more centers are needed for that respective input. A disadvantage of employing RBF and basis function in general is the extensive search space for a good parameterization. Think of designing a grid for the entire state space, where every input dimension requires a lower/upper bound and interval. This process results in a manual optimization in $\mathbb{R}^{3n}$, with n the input dimension. Suitable values for confined state-spaces are feasible for reasonable guesses, but for extensive state spaces this tuning process renders unfeasible.

2.4. Gaussian Processes (GPs)

In order to circumvent the extensive design procedure of RBF network, we introduce an alternative regression method: Gaussian Processes (GPs). In addition to the favorable engineering properties, a second advantage of this regression tool is the uncertainty estimates for predictions.

The section is organized as follows: the mathematical background of GPs is covered in Section 2.4.1. The covariance function, *hyperparameters* and corresponding effect on the regression will subsequently be detailed in Section 2.4.2.

2.4.1. Regression and uncertainty estimates

GPs are Bayesian Non Parametric function approximation models. It is a collection of random variables, such that every finite collection of those random variables has a multivariate normal distribution. GPs do not require a specification of the model structure a priori. The structure if fully dependent on the regression data. A GP is fully specified by its mean $m(\mathbf{x})$ and covariance function (i.e. kernel function) $k(\mathbf{x},\mathbf{x}')$, which are defined as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x},\mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned}$$

The kernel function $k(\mathbf{x},\mathbf{x}')$ is a measure of similarity between two input vectors $\mathbf{x}$ and $\mathbf{x}'$, and is an essential design choice in GP regression. We will cover this more extensively in Section 2.4.2. A regression performed by means of GPs is denoted as:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x},\mathbf{x}')).$$

Let $\mathbf{y} = \{y_1, \ldots y_n\}$ be a set of observations from a stochastic process

$$y_i = f(\mathbf{x}_i) + \epsilon,$$

where $\mathbf{x}_i$ denotes the input vector of observation y_i. For an intuitive understanding of GPs the observations $\mathbf{y}$ can be seen as a single point drawn from an n-variate Gaussian distribution. The noise ϵ is assumed Gaussian with standard deviation σ_o. The input matrix is defined as $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Applying the conditional distributions for GPs (Rasmussen and Williams, 2006), the following posterior predictive equations for test inputs X_* can be derived:

$$\begin{aligned} \mathbf{f}_*|X,\mathbf{y},X_* &\sim N(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*)), \quad \text{where} \\ \bar{\mathbf{f}}_* &\triangleq \mathbb{E}[\mathbf{f}_*|X,\mathbf{y},X_*] = m(X^*) + K_*[K + \sigma_n^2 I]^{-1}(\mathbf{y} - m(X)), \\ \text{cov}(\mathbf{f}_*) &= K_{**} - K_*[K + \sigma_n^2 I]^{-1} K_*, \end{aligned} \tag{2.9}$$

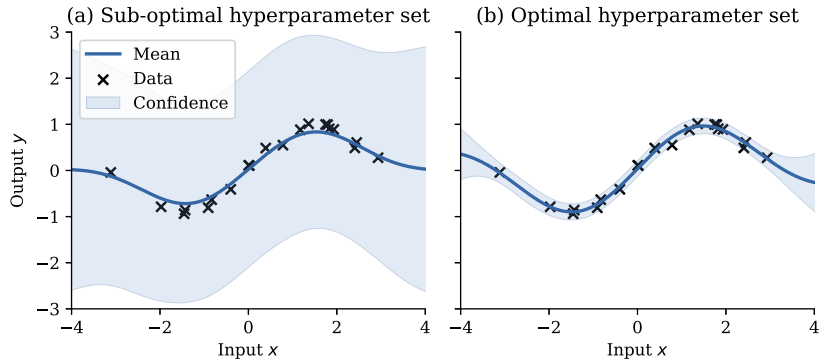


Figure 2.2: Example of optimization of the *hyperparameters* of a GP. In Fig. *(a)* and *(b)* a regression is performed on identical data points. In *(a)* the regression with sub-optimal hyperparameters values, in *(b)* a Bayesian optimized set.

where $K_* = k(X, X_*)$ and $K_{**} = k(X_*, X_*)$ as in

$$K_* = K(X, X_*) = \begin{bmatrix} k(\mathbf{x}_*, \mathbf{x}_1) & k(\mathbf{x}_*, \mathbf{x}_2) & \dots & k(\mathbf{x}_*, \mathbf{x}_n) \end{bmatrix}, \quad K_{**} = K(X_*, X_*) = \sigma_f^2 I + \sigma_n^2 I,$$

and K is the Gram matrix with entries $K_{ij} = k(x_i, x_j)$:

$$K = K(X, X) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix},$$

Note that the way regression is conducted by GPs has a lot in common with the employment of basis functions. Another way to look at (2.9) is by considering it as a linear combination of n kernel functions, i.e.

$$\bar{\mathbf{f}}_* = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*),$$

where $\alpha = (K + \sigma_n^2 I)^{-1}\mathbf{y}$ and $m(\mathbf{x}) = 0$. Note the similarity with (2.5). The foremost difference between GPs and feature spaces are the way the centers are mapped. Where feature space engineering requires a predetermined set of centers, the employment of GPs does not. The locations are dependent on the training data, which means that only no centers are found outside the relevant state space. Moreover, due to the fact that for every data instance a new center is introduced, the mathematical complexity of a GP hence scales with the amount input data. A consequence of employing GPs is therefore the poor scaling capabilities for substantial regression data. Observing (2.9) shows that for performing regression, the $n \times n$ inverse is needed of all measurements combined. Despite the fact that this matrix can be calculated off-line for predictions only, it is still a major drawback in an online application. A remedy is to sparsify the GPs (e.g. Csató and Opper (2002)), yielding more efficient predictions and uncertainty estimates at the costs of accuracy. Such sparsification techniques will be applied to where necessary in the remainder of this report.

2.4.2. Kernel and Hyperparameter properties

A major advantage of employing GPs is the ability to create confidence bounds on predicted outputs. The propagation of this confidence along the input boundaries is enclosed in the *kernel*. The choice of this covariance function depends on the data at hand. The main properties to be considered are the presence of discontinuities and the composition of the length-scales. For rough data a rather short length-scale has to be adopted, while slow and more smooth data should be accompanied by a longer length-scale.

There exists numbers of kernel functions that can be applied to GPs. In fact, any function that is symmetric and is positive semi-definite can act as a kernel function. However, In practice there are some kernels that are applied more generally. The first one is the Squared Exponential (SE) kernel, used to approximate smooth functions with no to little prior knowledge (See Duvenaud, 2014):

$$k_s(\mathbf{x},\mathbf{x}') = \sigma_c^2 \exp\left(-\frac{|\mathbf{x}-\mathbf{x}'|}{2l^2}\right),$$

with $\beta_r = \{\sigma_c, l\}$ the *hyperparameters* of the kernel function, consisting of the signal variance σ_c and length-scale l. The length-scale denotes a measure for the roughness of the data. In general, one can assume that extrapolating more than l units away from the input data is considered unreliable.

A kernel that allows for more freedom in modelling smoothness and (dis)continuities is the Matérn kernel:

$$k_m(\mathbf{x},\mathbf{x}') = \sigma_c^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{2}{l}\right)^{\nu} B_\nu \left(\sqrt{2\nu}\frac{|\mathbf{x}-\mathbf{x}'|}{l}\right),$$

with $B_\nu(\cdot)$ the modified Bessel function (Abramowitz and Stegun, 1965), $\Gamma(\cdot)$ the Gamma function and the *hyperparameters* $\beta_m = \{\sigma_c, l, \nu\}$. Here, ν denotes a 'smoothness' parameter that correlates with the amount of times the target function is differentiable (Rasmussen and Williams, 2006).

Every kernel and its *hyperparameters* shows different propagation of the uncertainty along the data dimensions. For a decent design, one has to examine the data at hand very closely and choose the kernel accordingly. The variance of the input-data itself does not have any influence on the estimated uncertainty. The estimated variance is in GP solely dependent on the sample density at a particular state, rather than the respective variance. A method to take the input-variance into account as well is by employing Heteroscedastic GP (see Le et al., 2005).

Once the kernel function is chosen, one has to tune the *hyperparameters* of the function, and in particular the length-scale. An example of its effect can be observed in Fig. 2.2. Two identical data sets with two kernel functions with different length-scales are approximated by a GP. In *(a)* we observe sub-optimal behavior with extensive (and presumably inaccurate) uncertainty. In contrast, *(b)* shows expected behavior and therefore uses an equitable values of the hyperparameters. For multivariate inputs one may adopt Automatic Relevance Determination (ARD) (Neal, 2012), which allows to determine an independent length-scale per input dimension. Tuning can be conducted manually or by optimization, e.g. Cross-validation or Marginal Likelihood (See Rasmussen and Williams, 2006; Wahba, 1990).

2.5. Conclusion

This chapter introduced the main concepts relevant to this study. In Section 2.1 Reinforcement Learning (RL) was covered. RL entails a learning framework where the policy is learned through trial and error. Iterative policy updates are conducted based on the obtained *return* during an episode. It typically takes a considerable amount of input-output data to achieve good performance: for instance a simple Atari game that takes human minutes to master, took up to 83 hours of experience for a RL algorithm to achieve similar performance (Hessel et al., 2018). The applications for RL are therefore mostly limited to simple problems or problems that can be modelled and simulated.

In Section 2.2, this inefficiency of training samples was addressed though the employment of the intuitive domain knowledge of a human. A human (or trainer) often possesses useful knowledge that can guide the agent to eventually converge to the optimal policy. Instead of exploring the whole state space, a learning problem can be accelerated significantly by limiting this. We have introduced algorithms that rely on demonstrations (in the LfD framework), but also algorithms that learning from evaluative and guidance feedback. The LfD paradigm has shown to require expert knowledge of the trainer (Argall et al. (2009)), which can hinder widespread application. Moreover, evaluations of humans have shown to implicitly contain guidance signals on future actions instead

of only evaluation of the past actions (Thomaz et al., 2006). Furthermore, according to Suay and Chernova (2011), evaluations have shown to have poor scaling capabilities to higher order systems. We therefore introduced the concept of directive feedback implementations and the current state-of-the-art: COACH by Celemin and Ruiz-del Solar (2015) in Section 2.3.

This algorithm engages RBF network to model the policy and the human model, and takes binary feedback to perform iterative updates. The disadvantage of this algorithm is the extensive parameterization procedure of the feature space in $\mathbb{R}^{3n}$, with n the input dimension. Since this process is done manual, it renders application for multi-state problems nearly unfeasible.

To improve upon the convenience of parameterizing the state-space, we have introduced an alternative to feature space engineering: GP in Section 2.4. GPs have a lot in common with the RBF network in COACH. The both entail regression with a linear combination of the features. The difference is that the centers of these features are predetermined in COACH, whereas a GP locates them at every training-instance. This introduces some advantages: 1) The ignorant input-space can be excluded from basis functions 2) It provides a natural way of obtaining the uncertainty. Moreover, instead of the engineering three parameters per input dimension in RBF networks (lower/upper bound and interval), a GP only need one (length-scale) per input dimension, which entails better scaling capabilities to multidimensional problems. The following chapter will explain how GPs can contribute to the learning process by presenting Gaussian Process Coach (GPC).

3

Gaussian Process Coach (GPC)

The current state-of-the-art regarding directional feedback implementations, COACH, suffers from extensive parameterization of RBF networks in higher order problems. In this chapter we will introduce an algorithm that will overcome this limitation by employing GPs as alternative to RBF networks. This novel framework, coined Gaussian Process Coach (GPC), will show to comprise advantages in terms of engineering convenience and feedback sample efficiency.

This chapter will present the algorithm in fullest detail. We will start with a comparison with COACH in Section 3.1, where the main differences and similarities will be covered. Section 3.2 will describe the architecture of the *agent*, consisting of a human (feedback) model and a policy. Section 3.3 will elaborate on the choice and parameters of the kernels of the GP models. Subsequently, Section 3.4 will explain how the uncertainty estimations will be leveraged: we will introduce an Adaptive Learning Rate (ALR) that will respond to the learning phase of the teacher. Moreover, the uncertainty estimates are used to inquire feedback and establish a pioneering approach in employing Active Learning (AL) in a directive feedback implementation. Lastly, the uncertainty is engaged for the design of a novel sparsification method specifically designed for iterative GP updates. The chapter will conclude with the total framework and the conclusion.

3.1. GPC versus COACH

The novel algorithm preserves COACH's structure in its architecture. A schematic of GPC is presented in Fig. 3.1. The policy is initialized idle. The trainer observes the environment and the actions of the agent. When the trainer observes an action that he would like to correct, he provides feedback $h \in \{-1, 1\}$ in the respective action dimension. The *human model* and the *policy* are updated immediately to observe the effect directly. This procedure, which is no different from COACH, is repeated until convergence.

The differences arise when looking a bit further into the technical details. First of all, the human and policy RBF networks are replaced by GPs, which avoid the extensive parameterization of feature spaces completely. Moreover, both GPs provide uncertainty estimations that are employed for 1) Communicating the uncertainty to the user to integrate AL and 2) Establish a learning rate that responds to the learning phase of the trainer. These two features are displayed in orange in Fig. 3.1. Since COACH does not allow for uncertainty estimations it is unable to leverage on such confidence bounds. If we compare Fig. 2.1 to Fig. 3.1 side-by-side, we can say that the human model of GPC combines the *Human feedback Supervised Learner* and *Human Feedback Modelling* of COACH, since GPC do not employ feature weight vectors that require updates by an external modules. The same accounts for the *Policy* and the *Policy Supervised Learner*, which are both integrated into the policy for GPC. Further details on the differences and similarities are elaborated in the following sections.

3.2. Modeling the human feedback and policy

Analogue to the COACH, GPC consists of two models that are contained in the agent (see Fig. 3.1): the policy P and the human model H. The prior of the policy is modelled as:

$$P : S \to \mathbb{R} \sim GP(m_p(\mathbf{s}), k_p(\mathbf{s}, \mathbf{s}')), \tag{3.1}$$

Here, $m_p(\mathbf{s})$ is assumed 0. The policy is trained with the set $N_p = \{(\mathbf{s}_1, a_1), (\mathbf{s}_2, a_2), ..., (\mathbf{s}_m, a_m)\}$, which contains state-action data derived from the directional feedback from the trainer (details in Section 3.4.3). The human feedback is modelled by

$$H : S \times A \to \mathbb{R} \sim GP(m_h(\mathbf{z}), k_h(\mathbf{z}, \mathbf{z}')), \tag{3.2}$$

with A the action space. The mean is assumed $m_h(\mathbf{z}) = 0$. This human model is trained with the set $N_h = \{(\mathbf{z}_1, h_1), (\mathbf{z}_2, h_2), ..., (\mathbf{z}_v, h_v)\}$, where $\mathbf{z}$ denote the concatenation of state $\mathbf{s}$ and action a, and $h \in \{-1, 1\}$ the suggested action correction of the teacher (decrease or increase). The proposed GPC introduces a different human model with respect to the one of COACH, where the feedback was only state dependent, i.e. $H_c : S \to \mathbb{R}$ (see Section 2.3). We have integrated the action in our human model to infer the human feedback per state-action, rather than state only. The associated uncertainty estimates therefore also reflect the uncertainty per action. Further details on this principle are provided in Section 3.4, where we elaborate on the corresponding uncertainty advantages.

Both models require a kernel function that represents how the target function and uncertainty propagates along the input dimensions, as explained in Section 2.4. For the human model H we assume a smooth propagation of the target function and therefore adopt the SE kernel. To allow for more freedom in the policy function in terms of roughness and continuities, we adopt the Matérn kernel for P (Rasmussen and Williams, 2006; Duvenaud, 2014). For the purpose of completeness and convience, the respective kernels are reprinted here. The SE kernel for the human model H:

$$k_s(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|}{2l^2}\right), \tag{3.3}$$

with hyperparameters $\beta_r = \{\sigma_s, l\}$. The Matérn kernel for the policy P is:

$$k_m(\mathbf{x}, \mathbf{x}') = \sigma_m^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{2}{l}\right)^{\nu} B_\nu \left(\sqrt{2\nu}\frac{|\mathbf{x} - \mathbf{x}'|}{l}\right), \tag{3.4}$$

with *hyperparameters* $\beta_m = \{\sigma_m, l, \nu\}$.

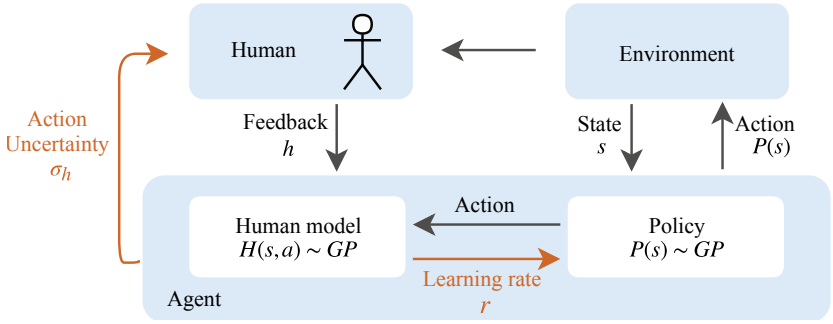


Figure 3.1: General structure of GPC. The human observes the environment and suggest action corrections ($h \in \{-1, 1\}$ for decrease and increase respectively) to the agent. The agent performs updates instantaneously on the human model and the policy for immediate effect. This process is repeated until convergence.

Algorithm 2 GPC Algorithm

1: **Given:**
   Kernels $k(\cdot)$ for human model H and policy P
   Hyperparameters β with M_{cs} or M_{ns}
   Constant learning rate c_r
2: **for** all k within episode **do**
3: Get state $\mathbf{s_k}$
4: Execute action $a_k = P(\mathbf{s}_k)$ with uncertainty $\sigma_p(\mathbf{s}_k)$
5: Obtain corrective advise $h \in \{-1, 1\}$
6: **if** $h \neq 0$ **then**
7: $\mathbf{z_k} = (\mathbf{s}_k, a_k)$
8: $h_{\text{est}} = H(\mathbf{z}_k)$ with uncertainty $\sigma_h(\mathbf{z}_k)$
9: Learning rate $r_k \leftarrow \sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r$
10: New action $a_n = a_k + r_k \cdot h_k$
11: Update dictionary P and apply SPARS(N_p)
12: Update dictionary H: $N_h = \{..., (\mathbf{z}_k, h_k)\}$
13: Update scaling $(M_p, M_h) \leftarrow \text{cov}(N_p, N_h)$ // NS only
14: Train GPs: TRAIN(P, H)
15: **end if**
16: **end for**

3.3. Kernel for GPC and GPC-NS

The policy in (3.1) and human model in (3.2) both concern a multidimensional regression on the input data. Each input dimension may however be subject to data with completely different orders of magnitude, such that a single length-scale is unsuitable. We therefore take an approach that allows us to set an independent length-scale per input dimension.

Let us consider the SE kernel from (3.3). Following Rasmussen and Williams (2006), the parameterization in terms of the *hyperparameters* results in

$$k_s(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T l_s M(\mathbf{x}-\mathbf{x}')\right),$$

with M the diagonal matrix consisting of the *characteristic length-scales* per axis. Such a covariance function implements ARD (Neal, 1995). This study adopts two distinct methods for determining the diagonal values of M. In the first approach we let the trainer decide on the respective relevance of the input dimensions:

$$M_{cs} = \text{diag}(\mathbf{w})^{-2},$$

with $\mathbf{w}$ a vector consisting of custom weights on the input dimensions. These values are determined a priori and deemed static throughout the learning process. This method is referred to as GPC(-CS). The second method concerns the normalization of the independent inputs for an equal relative dependency, resulting in an approach where any length-scale tuning is circumvented. The result is an approach that does not scale with the input dimension and could hence be decisive for higher-order systems. This parameterization is carried out by

$$M_{ns} = \text{diag}(\sigma_m)^{-2},$$

with σ_m the vector containing the variance of the independent input dimensions, which is updated for every feedback sample (see line 13 in Algorithm 2). This method will be referred to as GPC-NS.

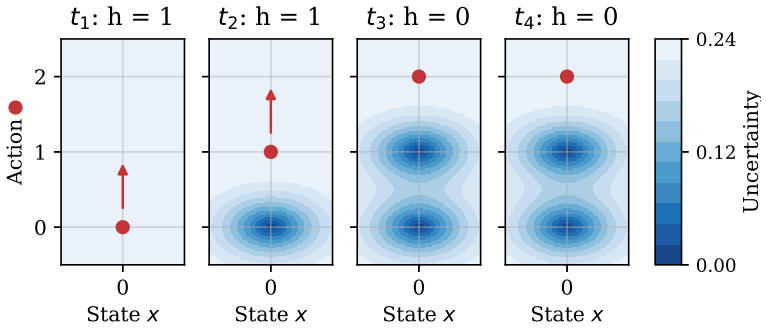


Figure 3.2: Hypothetical example of the propagation of the action for receiving 3 feedback instances. In t_1 and t_2 the trainers provides twice an increasing action correction. Consequently, the uncertainty is reduced for a (s,a)-pair and the action is increased. In t_3, when no feedback is received, nothing happens.

The extension to the Matérn kernel (3.4) is straightforward with $M_{cs} = \text{diag}(\mathbf{w})$ and $M_{ns} = \text{diag}(\sigma_m)$ for every length-scale l. To distinguish between the scaling of the policy and the human model we add subscript h and p, e.g. $M_{cs,h}$.

3.4. Leveraging uncertainty

GPs provide uncertainty estimates with every query point based on dissimilarity with the training data. For the policy, the uncertainty reflects the presence of feedback data in the respective or surrounding state. Due to the integration of the action in the input of the human model, this uncertainty reflects the presence of feedback for state-action pairs. To elaborate on this, a hypothetical example is depicted in Fig. 3.2. The contiguous plots show the evolution of the policy and its uncertainty as new feedback is obtained. At t_1 the policy is idle and the action corresponding to state $x = 0$ is zero. As an increasing action correction (i.e. $h = 1$) is received the action is increased as is shown from t_1 to t_2. The uncertainty of the old state-action concatenation has decreased due to this feedback sample. This process is repeated for t_2 to t_3. t_3 and t_4 show the consequence of receiving no feedback. We may envision this principle as building a map that discloses certain and uncertain regions with respect to past feedback. This feature comprises the main advantage of GPC over other methods.

This advantage is separated into three contributions, the Adaptive Learning Rate (ALR) in Section 3.4.1, AL in Section 3.4.2 and a novel sparsification technique for iterative GP policy updates in Section 3.4.3

3.4.1. Adaptive Learning Rate (ALR)

We assume that the teacher encounters two teaching phases during the learning period. The initial learning phase arises when the process is commenced and the policy is idle. We believe that the feedback in this stage will mostly concern raw adjustments in order to create a coarse version of the final policy. These coarse adaptations will gradually shift towards the second learning phase where trainers apply small refinements to the policy for meticulous improvements. In this study, we model the transition from coarse to fine adjustments not as a universal annealing process. Instead, we adapt the learning rate to the intended correction per state.

Hence, we introduce the following Adaptive Learning Rate (ALR):

$$r_k = \sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r, \tag{3.5}$$

with r_k the learning rate, $\mathbf{s}_k$ the state and $\mathbf{z}_k$ the concatenation of $(\mathbf{s}_k, a_k)$ (see line 7-9 in Algorithm 2). The uncertainty of the policy σ_p allows us to accelerate the learning by increasing the learning rate for the first feedback instances. The uncertainty estimation of σ_h adopts a high value for consistent feedback (see Fig. 3.2). As soon as alternating feedback is given, the uncertainty, and thus the

learning rate decreases to allow for refinements. The parameter c_r denotes the constant rate and prevents stagnation in the event that $\sigma_{p,h} \approx 0$. GPC differs from COACH for updating the policy, since the error magnitude e in (2.2) is now implicitly included in the computation of r_k in (3.5).

An example of the policy and the learning rate during a learning process is depicted in Fig. 3.3. It shows an environment with a two-dimensional continuous state-space and an unstable equilibrium as reference at $(x_1, x_2) = (0,0)$. The policy *(a)* is trained by a teacher employing the ALR. The corresponding learning rate is displayed in *(b)*. Note that for critical states (area around $(x_1, x_2) = (0,0)$) alternating feedback has caused the ALR to decrease, such that the policy can be refined.

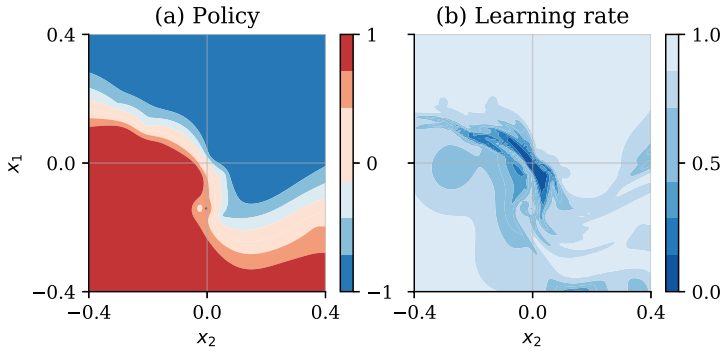


Figure 3.3: Snapshot of the policy *(a)* and learning rate *(b)* for controlling a system in an unstable equilibrium (0,0) (e.g. Inverted Pendulum). The learning rate decreased as a result of the ALR for alternating feedback that allows for meticulous action refinements in critical states.

3.4.2. Active Learning (AL)

The GPs used allow to actively elicit feedback in cases of high uncertainty in an Active Learning (AL) framework. Past studies have shown great performance improvement by enabling agent-induced feedback instances, mostly in the LFD domain (Losey and O'Malley, 2018; Gräve and Behnke, 2013). This study is, to the authors knowledge, the first to assess the potential with a directional feedback framework. Other than Chernova and Veloso (2009), who employed AL with the uncertainty of the visited state, we believe that the uncertainty of the action can advance the convergence in a directive feedback environment. The motivation for this reasoning is that, in contrast to the LFD paradigm, inquiring human assistance in terms of feedback does not yield the optimal action instantaneously. In GPC, multiple feedback instances are needed to approach the optimal action. Hence, rather than employing uncertainty per state, we apply uncertainty per action, which is obtained by the human model, i.e.

$$\Delta_k = c_a \sigma_h(\mathbf{z}_k), \tag{3.6}$$

with $\mathbf{z}_k$ the same as in (3.5) and a constant gain c_a to decouple AL from the ALR, due to the fact that in AL the relative differences between high and low uncertainty is relevant, rather than the magnitude. By inquiring feedback for high values of Δ_k we prioritize consistent feedback, since inconsistent feedback would reduce Δ_k. AL will therefore further aid in establishing an inaccurate but rather complete policy as early as possible, before proceeding to the refinement stage.

3.4.3. Novel Sparsification for Corrective Learning

For every feedback instance provided by the trainer, the dictionary of the policy P is appended with the new tuple:

$$N_p = \{..., (\mathbf{s}_{m+1}, a_{m+1})\} \tag{3.7}$$

where $(\mathbf{s}_{m+1}, a_{m+1})$ is calculated based on the executed action a_k, learning rate r_k and feedback h_k, i.e.

$$a_{m+1} = a_k + r_k \cdot h_k.$$

This approach renders the previous action a_k obsolete. A deficient property of GPs that hinders convergence is that by appending the dictionary following (3.7), the updated action on $\mathbf{s}_{m+1}$ is an average of a_k and a_{m+1} (assuming a coinciding or adjoining data instance). In this proposed sparsification method, the tuple most relevant to the obsolete action a_k is omitted, rendering a_{m+1} the new action. A visualization of this process for a specific state is depicted in Fig. 3.4. In *(a)* we observe the effect of appending the policy dictionary following (3.7): the action is the average of all data samples present. By removing redundant actions, we observe in *(b)* that that action follows the learning rate, as is desired.

This process, of taking the highest relevancy into account while preserving accurate uncertainty estimations has not been found in conventional online sparsification methods, (e.g. Nguyen-Tuong and Peters, 2010). We therefore introduce a new sparsification technique that specifically applies to applications with iterative updates on the GP policy model.

The main outline of this sparsification is as follows: for every new feedback instance $(\mathbf{s}_{m+1}, a_{m+1})$, the uncertainty of the policy $\sigma_p(\mathbf{s}_k)$ is compared against a certain threshold σ_{thres}, i.e.,

$$\sigma_{\text{thres}} = \frac{1}{2}\sqrt{\sigma^2_{s,m}},$$

with $\sigma^2_{s,m}$ from either (3.3) or (3.4). The uncertainty at a particular state measures the presence of neighbouring $(\mathbf{s}_k, a)$ tuples. In the event that this threshold is exceeded, the dictionary sample with the biggest covariance (i.e. smallest *Mahalanobis* distance (Mahalanobis, 1936)) is omitted. The sparsification method is presented in Algorithm 3 and executed simultaneously with appending N_p, see line 11 in Algorithm 2. The existing input elements of the policy dictionary N_p are denoted by $\mathbf{s}_i$.

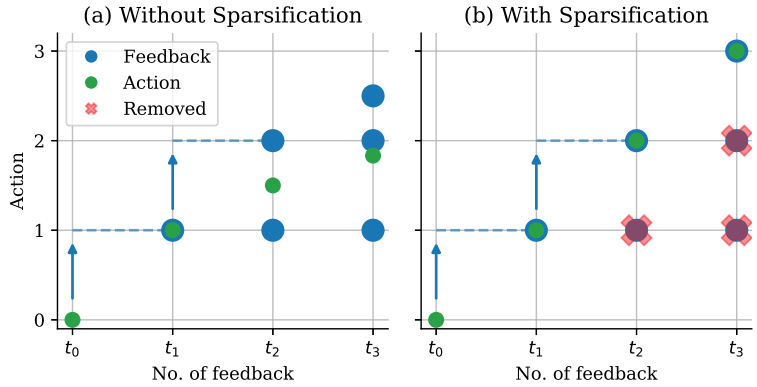


Figure 3.4: Sparsification of the Policy model P over time at a particular state. The consequences of repetitive feedback on the action is shown. *(a)* denotes the case where no old data is removed from the policy dictionary. *(b)* shows the removal of obsolete data, resulting in the action following the learning rate (blue arrow) seamlessly.

3.5. Framework GPC

The main outline of GPC is presented in Algorithm 2. Line 3-5 present the regular policy executions. As feedback is received, the updates in lines 6-14 are activated. The learning rate is composed of the uncertainty of both models, see line 9. Line 11-13 describe the dictionary updates and subsequent training of the GPs, which will result in an immediate effect on the policy.

3.6. Conclusion

The chapter detailed the novel directive feedback algorithm GPC. It has an architecture similar to the current state-of-the-art COACH, but comprises advantages by employing GPs instead of RBF networks such that no modelling expertise is not required anymore.

Algorithm 3 Sparsification of policy P training data

function SPARS($\mathbf{s}_k$, a_n, σ_p, σ_{thres}, N_p)
  if $\sigma_p < \sigma_{\text{thres}}$ **then**
    index $\leftarrow \arg\max_i \; k_p(\mathbf{s}_i, \mathbf{s}_k)$
    $N_p(\text{index}) \leftarrow (\mathbf{s}_k, a_n)$
  else
    Append dictionary $N_p = \{\ldots, (\mathbf{s}_k, a_n)\}$
  end if
  return N_p
end function

The engagement of GPs allows to obtain uncertainty estimates of the policy. We employ this uncertainty to 1) respond to the learning phase of the trainer by introducing an Adaptive Learning Rate (ALR) that decreased the learning rate as meticulous action corrections are required. 2) We furthermore establish a pioneering approach regarding AL in directive feedback frameworks. In contrast to the existing literature, which employ state-uncertainty for inquiring feedback, we apply the action uncertainty to trigger action corrections. 3) Lastly, we establish a novel sparsification technique for iterative updates of a GP policy. The next chapter will introduce the benchmarks to assess the significance of the introduced propositions.

4

Experimental Design

This chapter will detail the benchmarks for the novel algorithm GPC, and is organized by two main sections. The first section, Section 4.1, will explain *what* is going to be tested. This includes the performance and robustness of the algorithm, human validation and the AL feature. Section 4.2 details how this is tested. We will introduce three benchmark domain from the OpenAI gym (Brockman et al., 2016), which will be explained here. The chapter is concluded with a conclusion.

4.1. Experimental setup

This section will introduce the setup that is adopted for benchmarking the novel algorithms GPC. The experiments will be conducted with GPC, GPC-NS and baseline COACH. The performance is measured by means of the accumulated return per episode.

The section is organized as follows. First we will detail how the performance and robustness of all algorithms is measured. This is done in a fully customized setup by employing an oracle, a synthesized human, in Section 4.1.1. Section 4.1.2 will detail how the novel algorithm is validated for humans. The experiments regarding the significance of the AL and ALR proposition is subsequently described in Section 4.1.3 and Section 4.1.4 respectively.

4.1.1. Performance and Robustness

For the performance and robustness of the algorithms we employ an oracle. An oracle is a synthesized human that provides feedback, both ideal and erroneous, at a pre-specified rate. The employment of an oracle bypasses all human random factors, such as immeasurable inconsistencies and tiredness, and establishes a very neatly regulated setup that allows for fair cross-comparison between the algorithms.

An oracle measures the agent policy with a target policy. When the deviations is deemed significant, a feedback signal is provided to the agent. The agent policy is considered converged for a state when the deviation between target and executed action is δ close, and hence no feedback is provided. The target policy is pre-programmed by either a human, employing an interactive feedback algorithm like GPC, or by means of a conventional control algorithm (e.g. PID).

The performance of the algorithm is tested with a feedback rate of $\gamma = 5\%$. The robustness is assessed by an error e of 0%, 10% and 20%, and administered following the protocol of Celemin Paez et al. (2018), i.e., inverse the sign of h with likelihood e.

4.1.2. Human validation

This section will elaborate the experiments for validating the application of GPC to interactive settings. The experiments are conducted by employing four human teachers (in the age of 20 to 30, of

Algorithm 4 Oracle feedback

```
function FEEDBACK_ORACLE(s_k, a_k, Fb rate γ, error e)
    h = 0
    if rand(0,1) ≤ γ then
        a_o = optimal_action(s_k)
        if a_o < a_k − δ then
            h = −1
        else if a_o > a_k + δ then
            h = 1
        end if
        if rand(0,1) ≤ e then
            h =INVERSE_SIGN(h)
        end if
    end if
    return h
end function
```

different background) to the three proposed benchmarks with the objective to achieve convergence as fast as possible. The participants perform the training with every algorithm for every environment four times: two dummy runs to get acquainted with the environment, and two real runs that are recorded. The tests runs are performed single blind: the participants are not informed about which algorithm they controlled.

4.1.3. Active Learning (AL)

The potential regarding AL is assessed by encouraging feedback for uncertainty policy actions. In order to exclude any random human factors, the performance is measured using an oracle. As such, we will adapt the feedback rate by incorporating the uncertainty of the human model H, i.e.

$$\gamma = \Delta_k + \gamma_c,$$

with Δ_k as in (3.6) and γ_c denoting the minimum constant feedback rate. The application of AL is measured against a baseline with the same episodic feedback rate, but not uncertainty triggered (see Table 4.1, *ii* and *iv*). The ALR is excluded from the test to circumvent any influences. To account for feedback inconsistencies the erroneous feedback likelihood is set to $e = 10\%$.

Table 4.1: Overview of the experiments from Section 4.1.3 and Section 4.1.4. The significance of both Active Learning (AL) and the Adaptive Learning Rate (ALR) will be assessed. To account for feedback inconsistencies the error rate is set $e = 10\%$.

	AL:	ALR:	Fb rate γ:	Learning rate r_k:
i	✓	✓	$\Delta_k + \gamma_c$	$\sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r$
ii	✓	×	$\Delta_k + \gamma_c$	r_c
iii	×	✓	$\gamma_{\text{ep.avg}}(i)$	$\sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r$
iv	×	×	$\gamma_{\text{ep.avg}}(ii)$	r_c

4.1.4. Ablation Study

The ablation assessment will analyze the contribution of the ALR (Section 3.4.1). We will run oracle tests employing the learning rate in (3.5) and compare this to the baseline test from Section 4.1.3 with the same erroneous feedback likelihood. The episodic feedback rate γ is identical to i. In addition, we will test a combination of AL and ALR. A summary of the experimental setup from both Section 4.1.3 and this section is presented in Table 4.1.

4.2. Benchmarks

This section will introduce the benchmarks for testing the algorithms. The environments are all adopted from the OpenAI Gym (Brockman et al., 2016), which is a toolkit for testing and benchmarking RL algorithms. The algorithms adopted in this study are the Inverted Pendulum (Section 5.1.1), the Cart Pole (Section 5.1.1) and the Lunar Lander (Section 5.1.3), as depicted in Fig. 4.1. The environments are sorted with increasing complexity.

Figure 4.1: Snapshot of all domains adopted in this study. Most left the Pendulum-v0, in the middle the CartPole-v0 and on the right the LunarLander-v2. The domains are sorted with increasing complexity.

4.2.1. Benchmark 1: Underactuated Inverted Pendulum

The first domain concerns the Underactuated Inverted Pendulum, as depicted in Fig. 4.1 on the left hand side. The goal in this environment is to balance the pole upright with no velocity. The dynamics of the environment is governed by the following equation of motion:

$$\ddot{\theta} m l^2 + mgl \sin\theta = u(t),$$

where $g = 10\,\mathrm{m/s^2}$, $l = 1\,\mathrm{m}$ and $m = 1\,\mathrm{kg}$. The angle of the pole is denoted by θ. The environment counts three continuous state-dimensions: $\mathbf{x} = [\cos(\theta), \sin(\theta), \dot{\theta}]^T$, and is initialized randomly for every episode. Every episode takes 200 time-steps.

The pendulum domain employs a cost function rather than a reward function, i.e.,

$$C = (\theta - \frac{1}{2}\pi)^2 + 0.1\dot{\theta}^2 + 0.001\,u^2,$$

and consists of three parts: The first term denotes the deviation from the upright position, the relative velocity is represented by the second term and the third term accounts for the input costs. This latter is accompanied with a very low gain and has low contributions to the total costs. It is primarily used for fine-tuning in RL applied architectures, but will not be of major influence in this study. The costs are accumulated for all 200 timesteps.

4.2.2. Benchmark 2: Cart Pole System

The second domain concerns the Cart Pole environment as displayed in the middle in Fig. 4.1. It is slightly more complex compared to the inverted pendulum due to larger state-dimension. The

goal of the environment is to balance the pole on the cart by only actuating a force on the cart. The dynamics of the environment are governed by the following equations:

$$(M+m)\ddot{x} + mL\sin\theta\dot{\theta}^2 - mL\cos\theta\ddot{\theta} = u$$
$$mL^2\ddot{\theta} - mL\cos\theta\ddot{x} - mgL\sin\theta = 0,$$

where $M = 1\,\text{kg}$ and $m = 0.1\,\text{kg}$ denotes the mass of the cart and pole respectively. The (half length) of the pole is denoted by $L = 0.5\,\text{m}$ and $g = 9.8\,\text{m/s}^2$. The input is represented by u. The bounds of the independent states are given as $\pm 12°$ for the pole and $\pm 2.4\,\text{m}$ for cart position. The episode is terminated when one of the states exceeds the bounds, with a maximum of 2500 time-steps. All states are randomly initiated in the range $-0.05 > x_i > 0.05$ for every new episode. The continuous states are given by $\mathbf{x} = [x, \dot{x}, \theta, \dot{\theta}]^T$, with x the position of the cart and θ the angle of the pole.

The reward function for this environment is simple: for every timestep the reward is increased by one. The maximum accumulated reward is therefore equal to the time-limit and hence 2500.

4.2.3. Benchmark 3: Lunar Lander

The final domain comprises the Lunar Lander, presented in Fig. 4.1 at the right hand side. This environment is adopted in order to examine how the algorithms perform on more complex problems. The state-vector is given by $\mathbf{x} = [x, y, \dot{x}, \dot{y}, \theta, \dot{\theta}, \delta_1, \delta_2]$ and comprises both continuous and binary states. The first six states denote position, velocity and angular properties. δ_1 denotes a binary variable and is enabled whether the left leg is in contact with the ground. δ_2 accounts for the right leg.

The space ship needs to be landed between the two flags. The ship is always initiated at the top of the frame. The surface and the position of the flags are randomized every new episode. The difficulty in this environment is that the ship needs to be stabilized horizontally prior to landing on the landing pad. The first input a_1 is the main booster at the bottom to gain thrust into the direction of the ship. The second input a_2 controls the side boosters for rotation and translation. A value $-1 > a_2 > -0.5$ activates the left booster, whereas a value $0.5 > a_2 > 1$ activates the right booster. For $-0.5 > a_2 > 0.5$ the boosters are both not activated.

The reward for reaching the landing pad from the top of the screen is approximately 100 to 140 points based on the landing velocity. For landing between the flags an additional 100 is rewarded and for crashing -100. For firing the main engine the reward decreased by 0.3 each frame. Leg contact is awarded 10 per leg for the first contact.

4.3. Conclusion

This chapter introduced the experimental setup that is used to benchmark GPC. The novel algorithm will be measured against baseline COACH for overall performance, robustness and human applicability. The performance and robustness are measured by employing an oracle, a synthesized human. An oracle allows us to establish a very controlled feedback setting in which the rate and the error can be determined very accurately. The human applicability will subsequently be validated in confined experiments. An additional experiment is conducted to test the potential regarding AL. In this test, we increase the feedback rate of the oracle by the action uncertainty. The last experiment concerns the ablations study regarding the ALR, where we will assess its contribution. The experiments are ran on three different domains: the Underactuated Inverted Pendulum, the Cart Pole and the Lunar Lander, all from the OpenAI Gym (Brockman et al., 2016), a well-known benchmark utility for learning algorithms.

5
Results

This chapter will present the results for the experiments described in Chapter 4. We will first focus on the overall performance and the robustness of GPC compared to its baseline COACH in Section 5.1. Apart from the theoretical tests with oracles we will validate the applicability of GPC to human teachers in Section 5.2. The AL and ablation study is combined in Section 5.3. The results are finally discussed and concluded in Section 5.4 and Section 5.5 respectively.

Note that the results in this chapter will be shown with transparent and opaque lines (e.g. Fig. 5.1). The opaque lines are relevant to the respective figure. The transparent lines are shown to allow for cross-comparison between the figures.

5.1. Performance and Robustness

This section will present the results for the performance and robustness of GPC trained by a synthesized human. Each section will cover a different domain: the Underactuated Inverted Pendulum in Section 5.1.1, the Cart Pole in Section 5.1.2 and the Lunar lander in Section 5.1.3. The convergence properties will covered more extensively for the Pendulum domain since the limited state-dimension allows for good visualizations of the details.

5.1.1. Benchmark 1:Underactuated Inverted Pendulum

This section will detail the performance and robustness of GPC in the Underactuated Inverted Pendulum domain in the theoretical oracle tests. The adopted hyperparameters are depicted in 5.1 and are tuned by reasonable guesses and trial and error. For the target policy of the oracle we performed a training engaging the GPC framework. The tuning parameters for the baseline COACH are depicted in Appendix A.

Performance and Robustness

The return is presented in Fig. 5.1. We observe the most outspoken difference in the ideal feedback case *(a)* with 0% erroneous feedback. GPC and GPC-NS, which show identical behavior, outperform COACH in the leading domain of learning process by having a steeper learning curvature. GPC and GPC-NS have an advantage compared to COACH by having a coarser initial learning rate, as motivated in Section 3.4. In contrast, due to the *human feedback supervised learning* (see Celemin and Ruiz-del Solar (2015)) of COACH, a reduced learning rate is adopted for the leading learning domain, resulting in a slower convergence. The advantage of GPC over COACH is reduced by increasing erroneous feedback. for 20% in *(c)* we only observe a slight advantage of GPC over COACH.

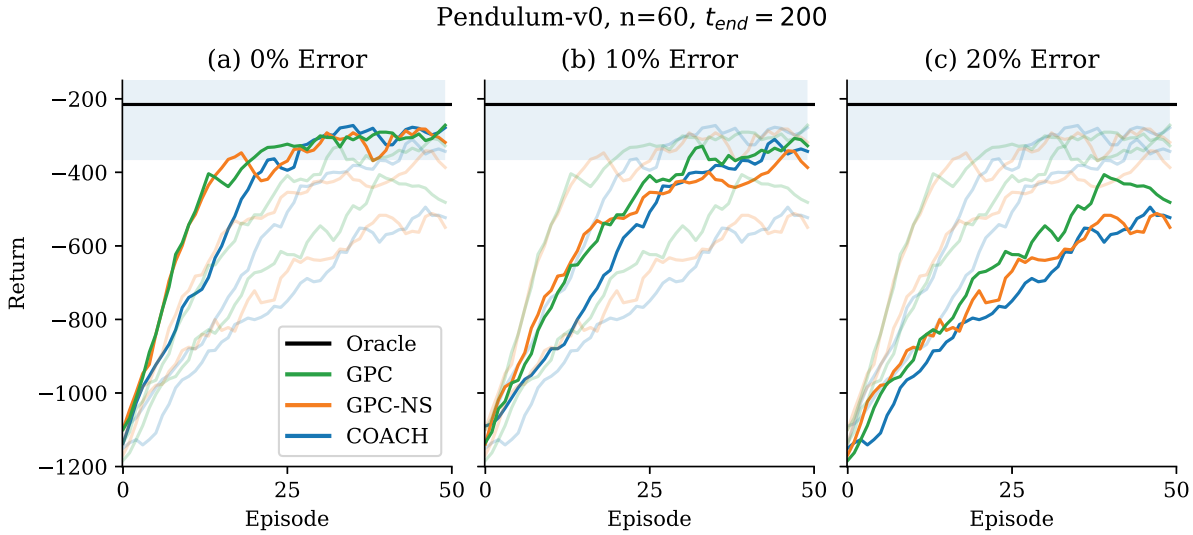


Figure 5.1: Performance of the three experiments with erroneous feedback in the Pendulum domain. 'Oracle' denotes the performance of the (synthesized) trainer, and is fit with a one standard deviation confidence interval. The return is depicted for $[0, 10, 20]$% erroneous feedback. The feedback itself is given at a constant rate of 5%. GPC and GPC-NS show similar performance and outperforms COACH in leading domain. Differences get less explicit for increasing erroneous feedback. The transparent lines are introduced to allow for cross-comparison between the respective figures.

Although the differences between the algorithms are not too outspoken in this domain, we will assess the convergence properties of GPC as the principles also extend to other domains where the differences will be more explicit.

Learning rate

The learning rate of the Pendulum domain is depicted in Fig. 5.2. The figure shows the learning rates for 10% and 20% erroneous feedback in order to make the relative differences more explicit. GPC shows in both figures the behavior we expect. Rough and coarse initial learning rate to create a raw version of the policy and a more subtle rate upon convergence to allow for meticulous corrections. Since for erroneous feedback the convergence is prolonged, the reduction of the learning rate is as well. For COACH the initial learning steps are small before consistent feedback causes the rate to increase, as can be seen in the figures.

State-space overview

A closer look on the converged GPC framework for 0% erroneous feedback is visualized in Fig. 5.3. We observe the converged policy in *(a)* and the individual contributions of the ALR for both the policy uncertainty $\sigma_p(\mathbf{s}_k)$ and the human model uncertainty $\sigma_h(\mathbf{z}_k)$ in *(b)* and *(c)* respectively. Interesting is the fact that the policy in *(a)* does not show the behaviour one would expect from a symmetrical problem like the Inverted Pendulum. This is presumably the result of the inconsistency of a human in its feedback. Additionally, a converged learning process from the trainer's point of view does not necessarily imply the overall optimal policy.

The learning rate in *(b)* induces the coarser learning phase in the leading domain of the learning process. In here, the dark spots mark the states that have received feedback during the learning process by having a reduced uncertainty. The light areas denote states without any feedback. The learning rate in *(c)* is decreased for alternating feedback, which allows for action refinements. In this figure $\mathbf{z}_k$ is concatenated of the receptive state $\mathbf{s}_k$ and the action a_k following the policy P. Note that the decreased uncertainty in *(c)* primarily occurs for $\dot{\theta} = \theta = 0$, i.e. where the pole is upright with no velocity. The alternating corrections causes the learning rate to diminish, where small corrections allow for an optimal fine-tuned policy.

The total learning rate is, referring to Table 2, composed of the sum of the learning rated depicted in *(b)* and *(c)*, plus an additional constant learning rate c_r. This latter component ensures a minimal value even when the learning rates induced by P and H stagnate and approximate zero.

Table 5.1: *Hyperparameters* for the Pendulum environment. See Section 2.4 for the properties of the hyperparameters. The difference between the algorithms is explained in Section 3.2.

	Pendulum	
	GPC	GPC-NS
Kernel H:	*SE*	*SE*
Constant Kernel c_h	0.7	0.45
Lengthscale l_h	0.1	0.1
Kernel P:	*Matérn*	*Matérn*
Constant Kernel c_p	0.01	0.03
lengthscale l_p	0.7	0.5
parameter ν_p	0.5	1.5
Learning Rate c	0.01	0.02
Scaling $\mathbf{w}$ for GPC	$\begin{bmatrix}1 & 1 & \frac{1}{6}\end{bmatrix}$	

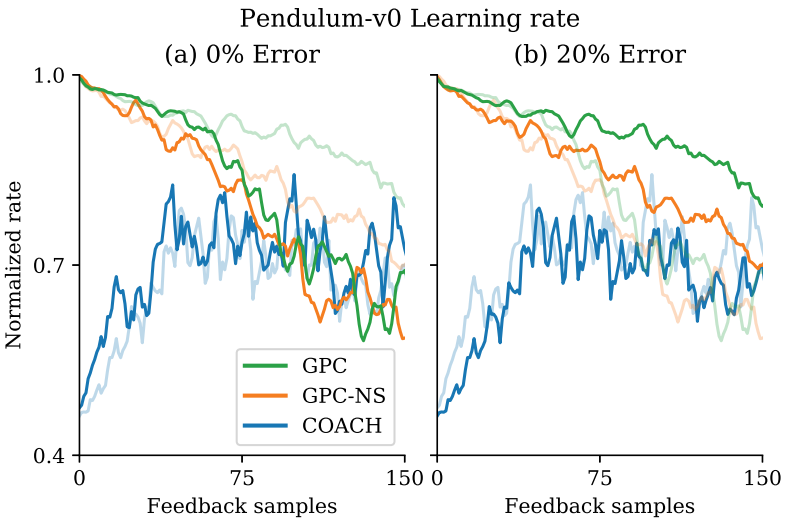


Figure 5.2: The average learning rate for the first 150 feedback instances of the learning process in the Pendulum domain, both for 0% and 20% erroneous feedback. We observe a reduced learning rate for GPC and GPC-NS upon convergence, which allow for policy refinements.

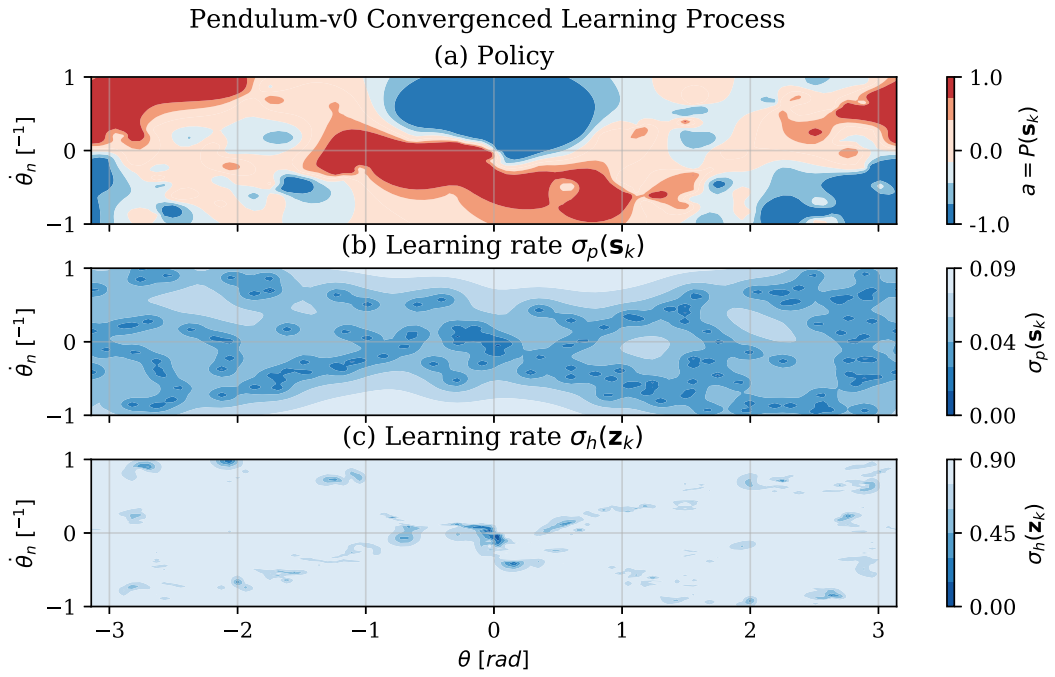


Figure 5.3: Convergence properties of the GPC algorithm for the Pendulum environment. *(a)* denotes the policy, with blue and red opposite action directions. For (b) and (c), the individual contributions of the components of the ALR (see $r_k = \sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r$ in (3.5)) are shown. *(b)* shows the learning rate induced by $\sigma_p(\mathbf{s}_k)$. The blue spots denote the areas where feedback has been given and where the learning rate hence dropped. In *(c)*, we observe the uncertainty for the human model $\sigma_h(\mathbf{z}_k)$. The dark areas mark a reduced learning rate for policy refinements.

5.1.2. Benchmark 2: Cart Pole

This section will detail the performance in the Cart Pole domain for the performance and robustness tests using an oracle. Again, the oracle is trained by a human employing GPC.

Performance and Robustness

The return for the oracle tests are depicted in Fig. 5.4 and concern the average return for $n = 20$ tests. The corresponding *hyperparameters* are displayed in Table 5.2. The parameters for baseline COACH are depicted in Appendix A.

The return shows a more explicit performance difference between GPC and baseline COACH. For ideal and erroneous feedback GPC is able to achieve superior and very robust performance. Analogue to the observed behavior in the Pendulum domain, we notice a steep initial learning curve of both GPC variants compared to baseline COACH. GPC-NS approaches the performance of GPC for ideal feedback, but shows, similar to COACH, brittle behavior for increasing inconsistent feedback. In Section 5.4 we will further detail the reason for this slightly sub-optimal performance.

Table 5.2: *Hyperparmeters* for the Cart Pole domain. See Section 2.4 for the properties of the hyperparameters. The difference between the different algorithms is explained in Section 3.2.

	Cart Pole	
	GPC	GPC-NS
Kernel H:	*SE*	*SE*
Constant Kernel c_h	0.01	0.08
Lengthscale l_h	0.2	0.5
Kernel P:	*Matérn*	*Matérn*
Constant Kernel c_p	0.01	10^{-3}
lengthscale l_p	0.2	0.7
parameter ν_p	1.5	1.5
Learning Rate c	0.02	0.05
Scaling $\mathbf{w}$ for GPC	$[1 \quad 1$	$\frac{1}{0.13} \quad \frac{1}{0.4}]$

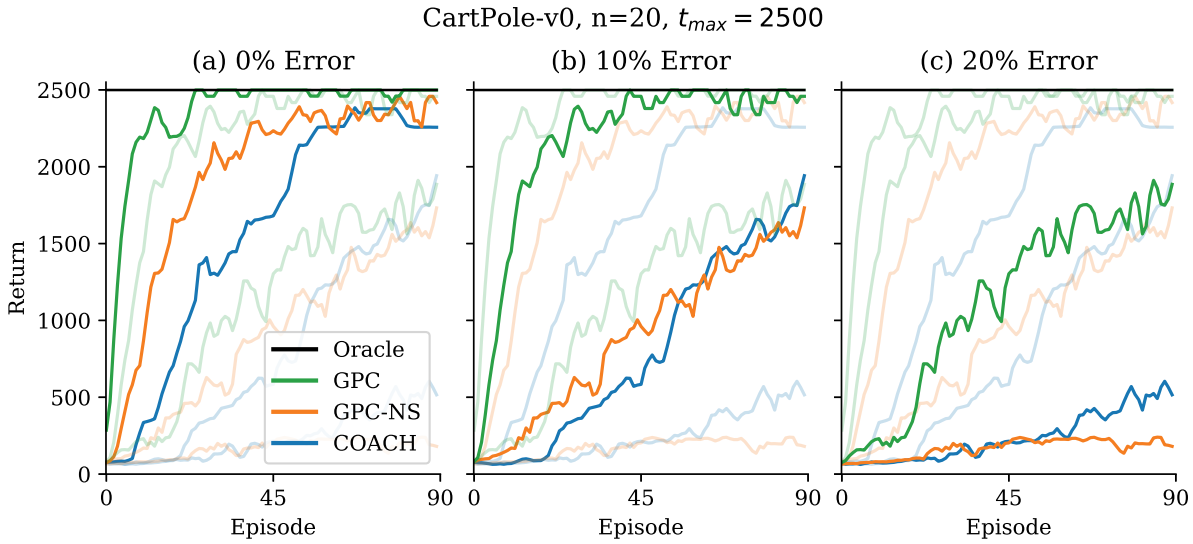


Figure 5.4: Return for the three experiments with increasing erroneous feedback in the Cart Pole domain. 'Oracle' shows the maximal obtainable return. Note the vast robustness of GPC to increasing erroneous feedback.

5.1.3. Benchmark 3: Lunar Lander

The last benchmarked environment is the Lunar Lander. The GPC *hyperparameters* are depicted in Table 5.3. The parameters for baseline COACH are presented in Appendix A. In contrast to the previous environment is the oracle not represented by a pretrained GPC policy. Instead, the target policy is generated by a PID controller, provided by the creators of the respective environment of the OpenAI Gym (Brockman et al., 2016).

Performance and Robustness

The return for the learning process with perfect feedback of the oracle can be observed in Fig. 5.5 *(a)* for $n = 100$ runs. The preset feedback rate of 5% was extended to two action-channels, resulting in 5% feedback for each action independently and therefore a total rate of 10%.

We observe good performance and robustness for both GPC variants. For all error rates a steep learning curve is shown that (presumably) converges on every tests. Interesting is the poor performance (and robustness) of the COACH algorithm. The state-space of the Lunar Lander consists of an eight dimensional continuous state space, which are all associated with an upper/lower bound

Table 5.3: *Hyperparameters* for the Lunar Lander environment, as explained in Section 2.4. The difference between the different algorithms is explained in Section 3.2.

	Lunar Lander	
	GPC	GPC-NS
Kernel H:	*SE*	*SE*
Constant Kernel c_h	0.01	0.08
Lengthscale l_h	0.2	0.2
Kernel P:	*Matern*	*Matern*
Constant Kernel c_p	0.01	10^{-3}
lengthscale l_p	0.4	0.6
parameter ν_p	1.5	1.5
Constant Learning Rate c	0.02	0.05
Scaling $\mathbf{w}$ for GPC	$[1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]$	

and an interval for designing the RBF network. This vast amount renders a manual determination, even in the case of educated guesses, unfeasible, and therefore results in poor performance. In contrast, GPC only requires a length-scale for every additional input dimension, and therefore has significant better scaling abilities to higher-order systems.

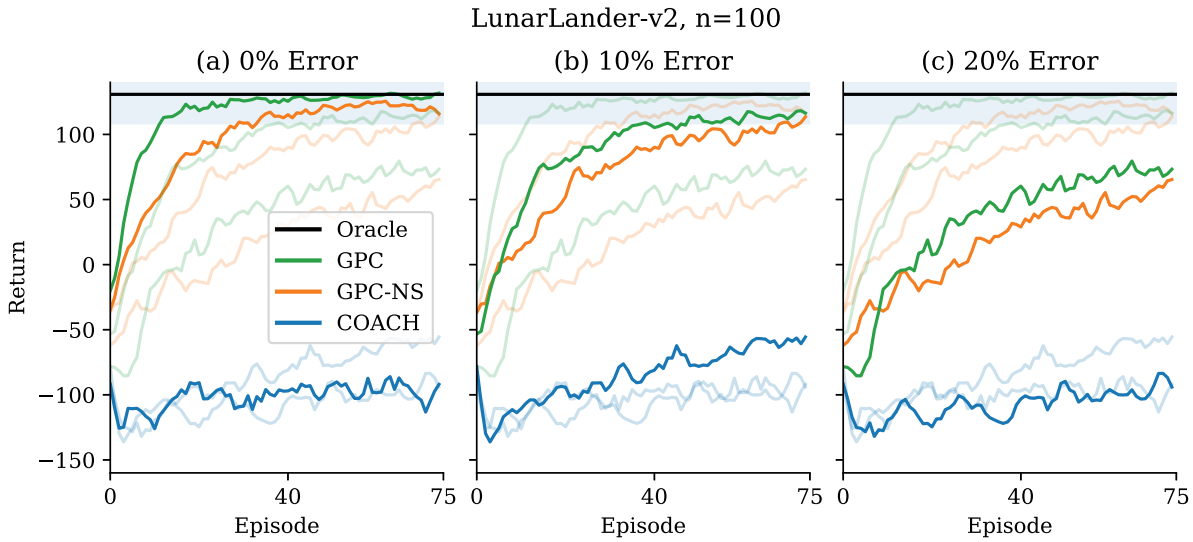


Figure 5.5: Return for the experiments in the Lunar Lander environment. COACH is unable to achieve any performance due to the infeasible parameterization of the feature space. GPC and GPC-NS show similar performance and achieve convergence on all error rates.

5.2. Human Validation

The following results present the applicability of the benchmarks to human teachers. The goal for adoption of these tests is to demonstrate that the algorithms can converge with humans and yield similar performance to the tests with oracles.

The results in all domains are depicted in Fig. 5.6. The tests are averaged over eight training sessions from four trainers with varying backgrounds following the protocol described in Section 4.1.1. The individual results are depicted in Appendix B. For the Inverted Pendulum and the Cart Pole environment both COACH and GPC seem to converge to maximal performance. Although some

relative differences are visible in the initial learning curve and final performance, the variations are not deemed statistically significant considering the number of tests. Interesting is the fact that the lack of robustness, that was present in Fig. 5.4 for COACH does not emerge from this result. This is probably due to the fact the human is better able to address possible shortcomings of a learning algorithm and adapts the feedback strategy accordingly, whereas an oracle is rather static and does not alter its feedback strategy to the observed behavior.

An interesting yet expected observation is the difference in return for the Lunar Lander environment in *(c)*. Despite the fact that humans might have been able to overcome the sub-optimal RBF network, they are still unable to achieve any performance for any run.

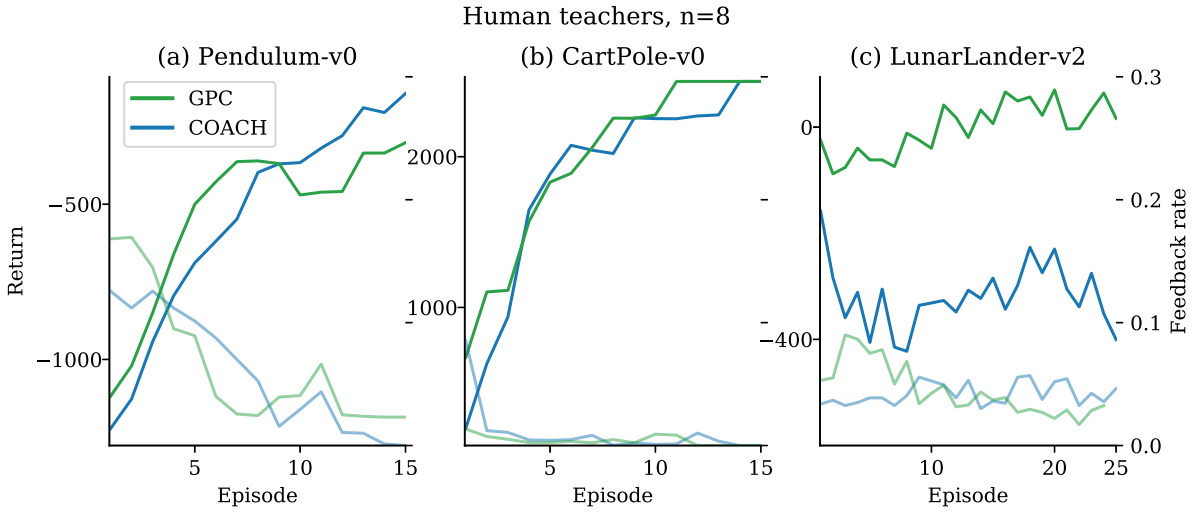


Figure 5.6: Return of the GPC and COACH algorithms on all environments tested in this study: *(a)* Inverted Pendulum, *(b)* Cart Pole and *(c)* Lunar Lander. The feedback was provided by four human teachers who initiated the learning process two times for every algorithm for every environment. The results are in line the oracle tests.

5.3. Active Learning (AL) and Ablation Study

In this section we will cover the contributions of AL and the ALR, as has been described in Section 4.1.4. The experiments were conducted in the Cart Pole domain, since the performance has shown the most explicit differences between COACH and GPC, whilst still attaining convergence for COACH. For the minimal feedback likelihood we adopted a value $\gamma_c = 0.01$ and a static learning rate of $r_c = 0.4$. For convenience purposes, the overview of Section 4.1.4 is reprinted here with the used values in Table 5.4. The return of the tests is shown in Fig. 5.7. The learning rate as a function of the feedback samples is depicted in Fig. 5.8.

First, let us focus on the advantages of AL. The comparison with the baseline learns us that inquiring feedback is very advantageous for the learning process. Both tests have the same feedback rate, but uncertainty induced for *ii* and randomly given for *iv*, which show to accelerate the convergence significantly. The convergence seems to benefit from the creation of a coarse policy first, where the refinements are postponed until later stages.

Engaging ALR in a learning process shows to increase the feedback sample efficiency and convergence rate. Compared to the baseline, enabling ALR reduces the required feedback, whilst still yielding superior performance. The reason for this can be observed in in Fig. 5.8. The ALR shows to balance the coarse and the refinement stage better, resulting in bigger learning steps in the leading domain, and reduction of the step size upon convergence. Note that the average (AVG) learning rate for the ALR is slightly less than for the static learning rates.

Superior overal performance is achieved for the combination of AL and ALR. This version, em-

ploying both features, requires the least feedback and attains the highest return of all experiments. Interesting is the slower decreasing learning rate curvature in Fig. 5.8 compared to ALR only in *iii*. This relative difference is due to symbioses between AL and the ALR: the action uncertainty is embedded into both the learning and feedback rate, resulting in a positive correlation and thus a higher chance of feedback with higher learning rates.

Table 5.4: Reprinted overview of the experiments regarding AL and the ablation study, where the independent contributions of both propositions are assessed.

	AL:	ALR:	Fb rate γ:	Learning rate r_k:
i	✓	✓	$\Delta_k + 0.01$	$\sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + 0.1$
ii	✓	✗	$\Delta_k + 0.01$	0.04
iii	✗	✓	$\gamma_{\text{ep.avg}}(i)$	$\sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + 0.1$
iv	✗	✗	$\gamma_{\text{ep.avg}}(ii)$	0.04

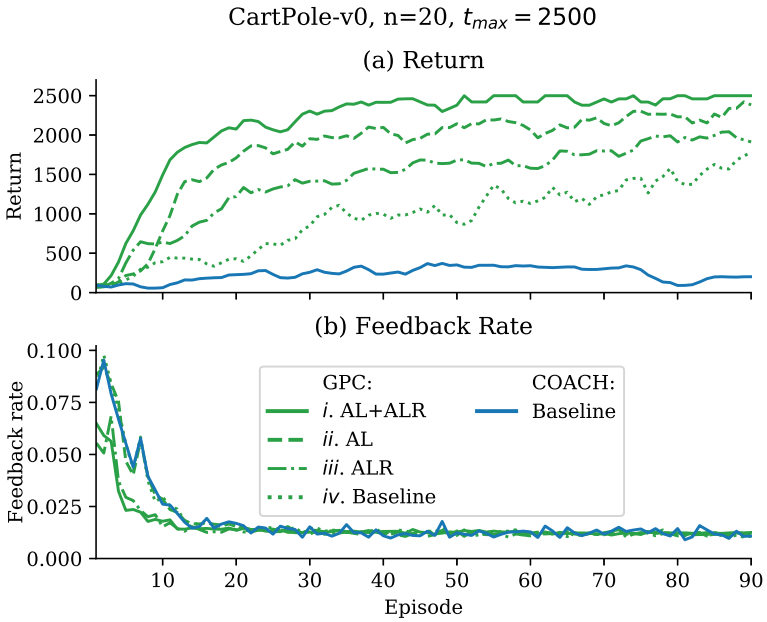


Figure 5.7: The performance of Active Learning (AL) and Adaptive Learning Rate (ALR) experiment. Both features combined shows superior performance, even though the least feedback was received. The significance of the independent components is showed by comparing the baseline to respective AL and ALR. Both features show to accelerate the convergence significantly.

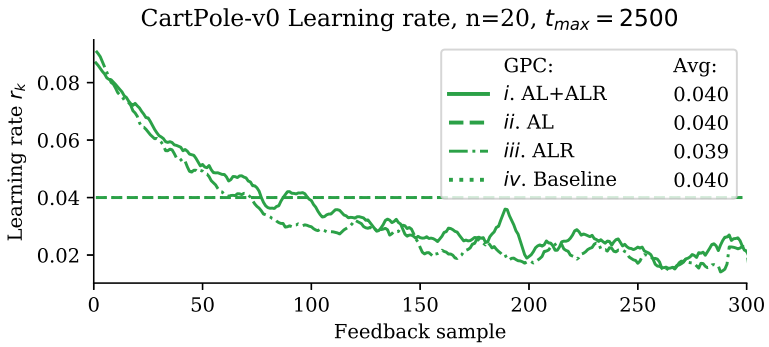


Figure 5.8: The learning rate as a function of the feedback samples from the experiments in Fig. 5.7. The static learning rates for AL and the Baseline is 0.04. The ALR-engaged experiments establish a better balanced learning rate which is even lower on average.

5.4. Discussion

This section will discuss the results from earlier sections. We will assess the advantages and disadvantages of GPC and COACH and drawn conclusions for applicability and generality. The overall performance and robustness is discussed in Section 5.4.1. The results of the AL and ALR ablation study are covered in Section 5.4.2. The side-to-side comparison of GPC and GPC-NS is done in Section 5.4.3.

5.4.1. Performance and Robustness

This study covered the assessment of GPC and GPC-NS, a variant to GPC with a dynamical scaling of the input data. Both variants have shown to outperform the current state-of-the-art on the majority of the domains. In particular GPC, which performance was never exceeded by any other algorithm. The most outspoken difference to the current state-of-the-art arose in the Lunar Lander domain, the most complex problem adopted in this study. Both GPC variants showed to fast convergence, whilst COACH was unable to achieve any performance. The reason for lack of performance is that COACH has a major disadvantage when it comes to parameterizing the RBF network. This parameterization process requires three dimension scales per input state, which renders unfeasible for higher order systems. Reasonable guesses for the respective numbers are complex, since assumptions about the input bounds and roughness are required for applicable numbers. In contrast, GPC requires only one length-scale per input dimension and therefore employs better scaling abilities to higher order systems. Furthermore, GPC-NS does not scale at all with the amount of inputs. The input-vector is normalized in their respective length-scales, rendering equal dependencies for every state-dimension. GPC and GPC-NS have shown to have greater scaling advantages, with superior scaling abilities for GPC-NS.

However, while GPC-NS shows to be best scalable in terms of tuning parameters, it has also demonstrated to have brittle robustness in the Cart Pole domain. This is, on one hand, presumably partly due to the sub-optimal input scaling of the regression data, but the combination with the *hyperparameters* could have amplified this effect. Observe Fig. 5.9, where the effect of an erroneous feedback sample is displayed for two hyperparameters sets. Set 1 has a small length-scale for the human model H whilst set 2 has a large length-scale. For the first set we observe desired behavior: an inconsistent sample is easily compensated (orange arrow). In contrast, the erroneous sample in set 2 shows to dwindle the learning rate due to a sudden decrease of $\sigma_h(\mathbf{z}_k)$ and $\sigma_p(\mathbf{s}_k)$ as a result of the large length-scale. This undesired behaviour of stagnation can be easily compensated by adopting a shorter length-scale for the human model, but might be at the costs of performance for ideal feedback. The tuning process regarding the GPC-NS algorithm for the Cart Pole environment was due to this reason a trade-off between performance for ideal feedback and robustness to erroneous

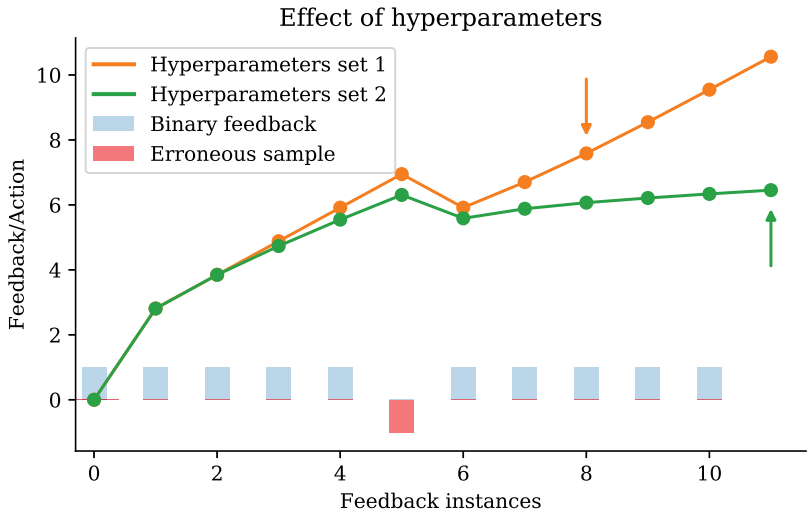


Figure 5.9: Effect of the hyperparameters on the development of the action as a function of feedback instances. An optimal set compensates for erroneous feedback quickly (*set 1*), whereas a sub-optimal stagnates into regions of low uncertainty and therefore needs multiple feedback samples for compensation (*set 2*).

samples, where the former was prioritized.

The validation with human teachers showed roughly the behavior that was in line with all oracle tests. Two interesting observations were done. The first observation concerned the brisk robustness of COACH in the Cart Pole environment, which did not arise in the human tests. Assuming that human do provide inconsistent feedback, this effect is presumably the result of the human teachers being able to respond to the consequences of their feedback, and are therefore able to compensate for any algorithm shortcomings, in contrast to an oracle. The second interesting observation was the inability for humans to converge Lunar Lander with the COACH algorithm. Due to reasons given earlier this section, does COACH suffer from an unfeasible tuning process regarding the RBF network and therefore attains no performance at all. This example shows the main proposition of GPC over other methods.

5.4.2. Active Learning (AL) and Adaptive Learning Rate (ALR)

In this study we have covered two features that were introduced to advance the learning properties of GPC: AL, where a pioneering approach has been conducted regarding directional feedback algorithms. And ALR, where the learning rate is adapted to the learning phase of the teacher.

The action uncertainty, obtained from the human model H, has shown to be an efficient measure to mark states that have not converged to the optimal (targeted) action. By increasing the oracle likelihood for feedback by the uncertainty of the human model H we have seen a significant increase in performance. However, the pioneering approach in this study does not prove the actual applicability to real human teachers. AL has shown potential in the adopted setting, however, the extension to real trainer is a rather complicated procedure. This application would involve the incorporation of many (for now) unknowns: like a good way of communicating the uncertainty to humans. Moreover, a delay has to be taken into account to compensate for the time it takes for the trainer to interpret the uncertainty together with the state and corresponding feedback. Therefore, further examination to the human applicability should be conducted to conclude on this.

The second feature we have tested concerns the ALR. For the ALR we have shown that the combination of policy uncertainty and human model uncertainty following (3.5) advanced the convergence of the algorithm significantly. The proposed learning rate establishes a balanced trade-off between a coarse initial policy creation and meticulous adaptions upon convergence.

5.4.3. GPC versus GPC-NS

We have presented two variation of the newly proposed algorithm: The standard version of GPC and its derivative GPC-NS. Whilst GPC has shown to outperform baseline COACH on every domain tested, GPC-NS attained performance that was able to approach GPC, but showed sub-optimal behavior on selected problems for erroneous feedback.

GPC concerns a method that does not employ RBF networks and therefore include advantageous scaling properties for multidimensional systems. Where RBF networks parameterize the state-space with three values per input dimension, GPC only needs one. Compared to COACH, GPC attains higher performance while requiring less tuning parameters. A step further is the introduction of GPC-NS. The difference with GPC is minor: the input data is normalized instead in stead of manually pre-processed. The main innovation is that GPC-NS therefore does not scale with the amount of input dimensions, establishing a very lean algorithm that can nearly be applied to any domain straight out of the box. However, every benefit has a cost. The automatic relevance scaling of GPC-NS shows to be sub-optimal on selected problems compared to GPC. The dynamical scaling establishes a very scalable algorithm, but it is to the trainer to decide on which algorithm to apply.

5.5. Conclusion

This chapter presented the results of the experiments depicted in Chapter 4.1. The novel algorithm, GPC, was measured against baseline COACH on three different benchmarks: the Inverted Pendulum, the Cart Pole and the Lunar Lander environment.

The first tests concerned the assessment of the overall performance and the robustness of the algorithms. These experiments were carried out by an oracle, a synthesized human. The results show that GPC is able to outperform baseline COACH on every domain tested. Interesting yet expected was the performance advantage of GPC in the Lunar Lander domain compared to baseline COACH. COACH employs RBF networks, which are engineered by tuning a grid over the input space, requiring a lower bound, upper bound and an interval per input dimension. This process results in a parameterization in $\mathbb{R}^{24}$. Despite reasonable guesses, the tuning space dimension renders this process unfeasible. Alternative to GPC, we have benchmarked GPC-NS, a version that performs automatic input scaling and therefore employs optimal scaling abilities to higher order systems. It has shown to approach the performance of GPC but appears to be sup-optimal on the Cart Pole domain for increasing inconsistent feedback.

The performance and robustness results were validated for by human teachers. For these tests, four trainers performed two runs on each adopted domain. The results were in line with what was expected from the oracle tests. Despite the fact that human are able to anticipate better to algorithm shortcomings, no performance was attained for COACH in the Lunar Lander domain.

The results were concluded by the AL and ALR assessment. We have seen a significant improvement of the convergence rate for both AL and ALR independently. The ALR balances the learning rate very neatly, increasing sample efficiency and obtaining faster convergence. The action uncertainty has shown in AL that it can be used to converge faster by inquiring feedback for high uncertainty. Note that this study only provided a pioneering approach regarding AL. The actual human application is still a very interesting topic.

6

Conclusion and Recommendations

The main objective in control engineering is to find a state-action policy that drives the system to a certain reference. A prevalent model-free approach to derive a control law is the Reinforcement Learning (RL) domain. Past studies (e.g. Hessel et al. (2018); Sutton and Barto (2018)) however show that this technique requires extensive interaction with the environment and therefore renders real-world applications unfeasible. In contrast, humans are very effective in inferring suitable control strategies when facing new problems. Specifically for intuitive problems, like picking up objects or playing simple games, humans are able to achieve decent performance on first try (Hessel et al., 2018). Communicating this domain knowledge has shown to drastically accelerate model-free control techniques. A very recent and innovative development is the introduction of directive feedback frameworks, which show to be very effective on intuitive problems without requiring expert knowledge on the modelling and control. The current state-of-the-art regarding directional learning was introduced by Celemin and Ruiz-del Solar (2015) and coined COrrective Advice Communicated by Humans (COACH). COACH takes action corrections from the trainer to update the policy in an iterative fashion. It shows to outperform conventional evaluative feedback algorithms on confined problems. However, COACH employs Radial Basis Function (RBF) networks, which have poor scaling abilities to higher order systems due to the vast amount of tuning parameters.

To improve on this point, we introduced Gaussian Process Coach (GPC). This algorithm preserves COACH's structure, but employs Gaussian Processes (GPs) as an alternative to RBF networks. It allows us to utilize the available uncertainty in three ways. 1) The learning rate is adapted to the teacher's learning phase in the Adaptive Learning Rate (ALR). 2) We provide a pioneering approach regarding the application of Active Learning (AL) to directive feedback implementations. 3) We introduce an uncertainty based novel sparsification technique that specifically applies to iterative GP policy updates.

The novel algorithm was tested on three OpenAI Benchmarks (Brockman et al., 2016) for various performance measures. The conclusions in the next section, Section 6.1. Suggestions on further development are provided in Section 6.2.

6.1. Conclusions

The contributions of GPC can be summarized as follows:

- **GPC has better scaling abilities compared to COACH** GPC employs GPs as an alternative to RBF networks of COACH. For every additional input parameter, a GP requires an additional length-scale to be set. In contrast, RBF networks require 3 extra parameters per input dimension, rendering parameterization unfeasible for higher order problems. Employing GPs in

the GPC framework has shown to be a suitable alternative to RBF networks, preserving fast performance whilst requiring less tuning inconvenience.

- **AL can accelerate the learning of GPC** The action uncertainty estimate from the human model H infers about the confidence of the executed action. For uncertain regions, the agent can ask the trainer for feedback in order to converge faster. A pioneering approach regarding AL has shown that inquiring feedback samples for high uncertainty advances the learning properties significantly. A side-note is that these tests were executed with an oracle (synthesized human) only. The human validation is an encouraged topic of research.

- **The ALR accelerates convergence of GPC** This study has employed an ALR, that establishes a coarser learning rate for the leading domain which diminishes upon convergence. Ablation tests have shown that this transition accelerates the learning drastically by establishing a balanced rate per state.

- **GPC-NS is a suitable alternative to GPC** This study introduced an alternative version of GPC: GPC-NS. The difference with GPC is minor: GPC-NS employs automatic scaling of the input data and therefore does not scale with the state-dimension of the problem. This feature could be decisive for higher-order systems where no extra tuning is required in contrast to other methods. GPC-NS showed to approach GPC's performance and robustness on the majority of the experiments. However, some slight sub-optimal behavior was shown on the Cart Pole environment to erroneous feedback. It is therefore up to the trainer and the application to decide on the variant to employ.

6.2. Recommendations

Next to the conclusions that have been a result of this study, the potential of GPC and directional feedback in general could be researched more in the following potential fields:

- **Human validation of AL** The foremost recommendation for future research is the study for active learning. GPs have shown to be an effective manner of estimating the uncertainty of actions and trigger feedback accordingly to accelerate the learning process. However, the human validation, including all factors that could be of influence, is still an interesting topic for research.

- **Further innovate dynamical scaling of GPC-NS** It might be possible to further innovate the dynamical scaling, such that the applicability and generality of GPC-NS is again extended and shows robust behavior on all domains. Moreover, the setting with autonomous scaling is yet to be validated on humans to conclude about physical applicability.

- **Limitations GPC** GPC offered a suitable alternative to RBF networks by introducing GPs. However, a well known disadvantage is the poor online scaling abilities to vast amounts of data. It would be interesting to know the limitations of GPC and ways to overcome this.

A

COACH Parameters

In Table A.1 the tuning parameters of the COACH framework are depicted for the adopted domains. The RBF networks requires 3 parameters per input dimension n, which does not scale for higher order problems, as has been detailed in Section 2.3.2.

Table A.1: Parameters of the COACH algorithm for all adopted environments. The meaning of the symbols is explained in Section 2.3. The upper and lower bound of the feature space are denoted by ub. and lb. respectively. The amount of intervals is represented by int.

COACH Parameters									
	Pendulum			Cart Pole			Lunar Lander		
COACH:									
Learning rate α		30			30			$\begin{bmatrix}1 & 1\end{bmatrix}$	
Error e		5			5			$\begin{bmatrix}5 & 1\end{bmatrix}$	
Constant learning rate c		1.8			1.8			$\begin{bmatrix}1 & 20\end{bmatrix}$	
Features		RBF			RBF			RBF	
Feature space:	lb.	int.	ub.	lb.	int.	ub.	lb.	int.	ub.
State 1	-1	6	1	-1.44	4	1.44	-1	2	1
State 2	-1	6	1	$-.12$	4	.12	-1	3	1
State 3	-6	7	6	$-.126$	4	.126	-1	2	1
State 4				$-.4$	4	.4	-1	2	1
State 5							-1	2	1
State 6							-1	2	1
State 7							-1	2	1
State 8							-1	2	1

B

Individual results Human teacher

Individual results of the experiments with human teachers. For the Underactuated Inverted Pendulum in Appendix B.1, for the Cart Pole in Appendix B.2 and for the Lunar Lander in Appendix B.3.

B.1. Underactuated Inverted Pendulum

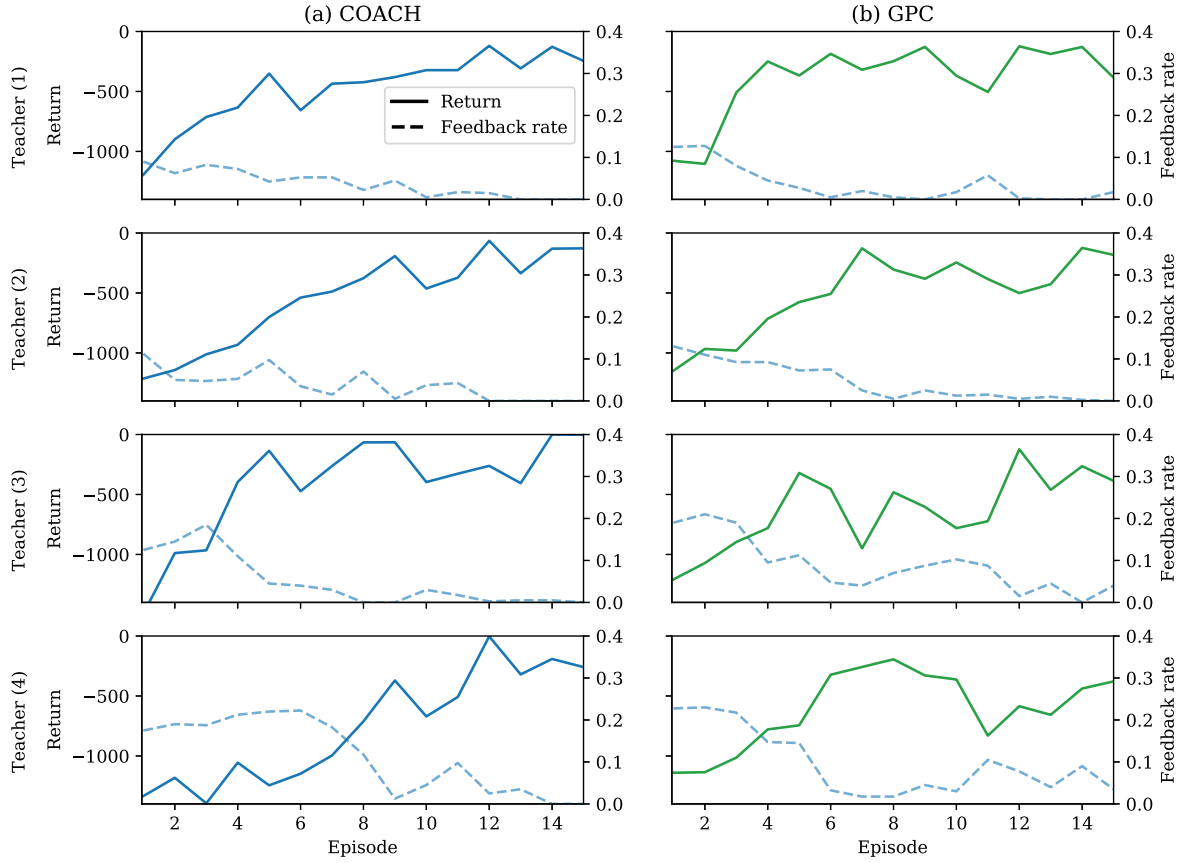


Figure B.1: Individual results for the Underactuated Inverted Pendulum environment for the experiments with human teachers. The rows distinguish the 4 trainers.

B.2. Cart Pole

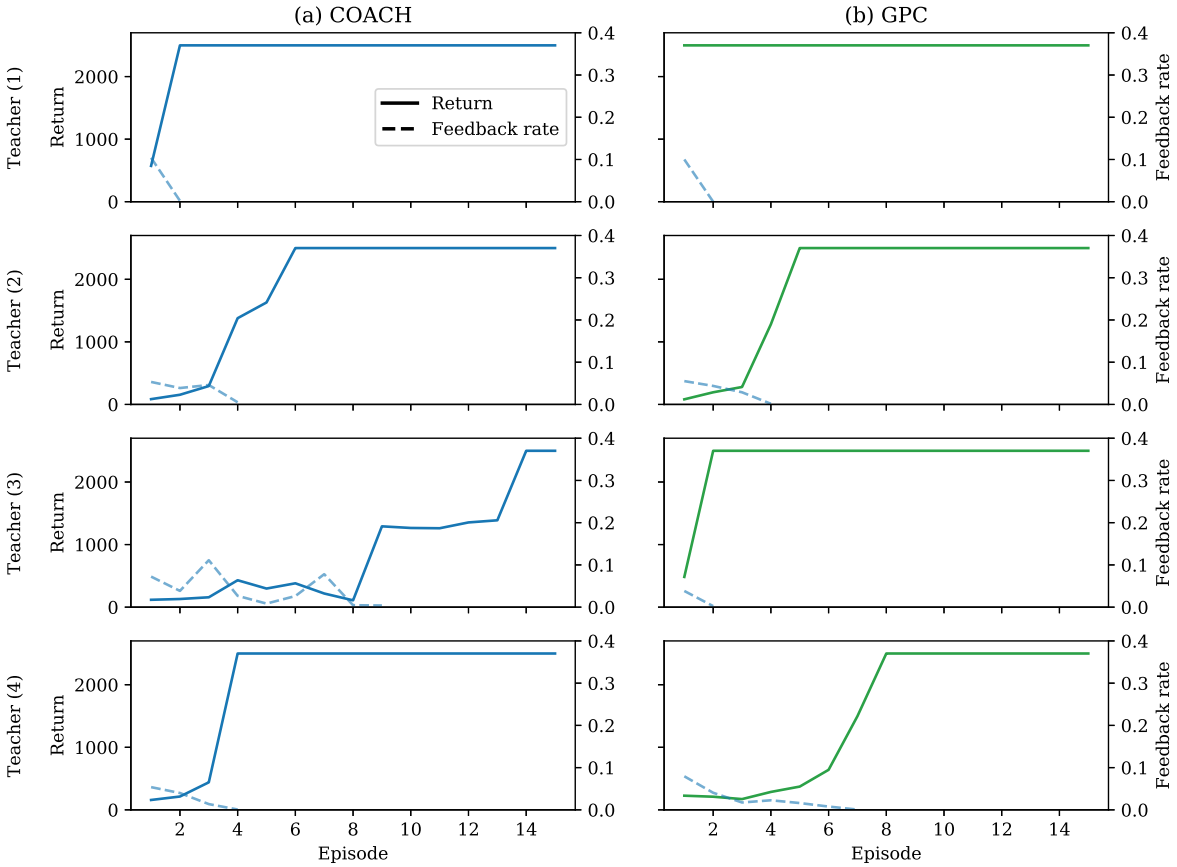


Figure B.2: Individual results for the Cart Pole environment for the experiments with human teachers. The rows distinguish the 4 trainers.

B.3. Pendulum

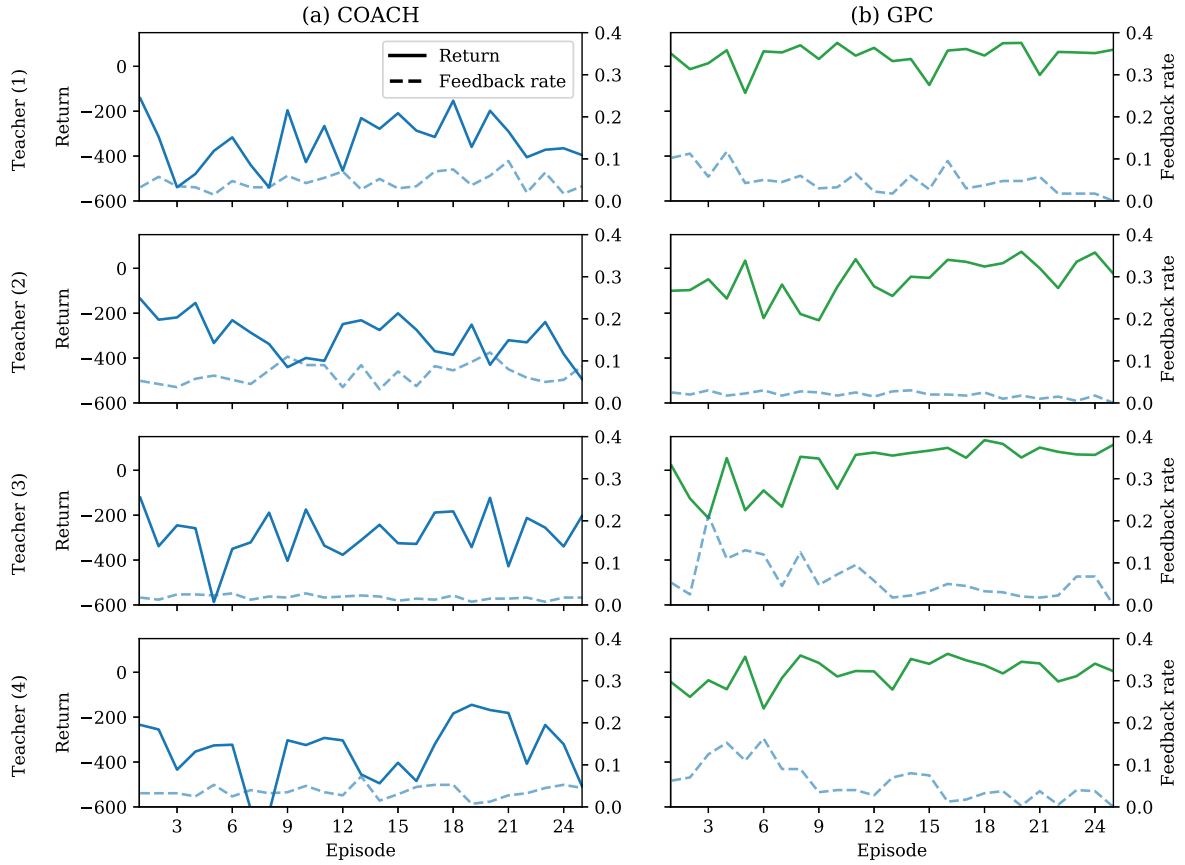


Figure B.3: Individual results for the Lunar Lander environment for the experiments with human teachers. The rows distinguish the 4 trainers.

C

Paper

This paper is submitted to the 2019 Conference on Uncertainty on Artificial Intelligence, see http://auai.org/uai2019/, and is uploaded to arXiv with ID 1903.05216.

Learning Gaussian Policies from Corrective Human Feedback

Daan Wout* **Jan Scholten** **Carlos Celemin** **Jens Kober**

Abstract

Learning from human feedback is a viable alternative to control design that does not require modelling or control expertise. Particularly, learning from corrective advice garners advantages over evaluative feedback as it is a more intuitive and scalable format. The current state-of-the-art in this field, COACH, has proven to be a effective approach for confined problems. However, it parameterizes the policy with Radial Basis Function networks, which require meticulous feature space engineering for higher order systems. We introduce Gaussian Process Coach (GPC), where feature space engineering is avoided by employing Gaussian Processes. In addition, we use the available policy uncertainty to 1) inquire feedback samples of maximal utility and 2) to adapt the learning rate to the teacher's learning phase. We demonstrate that the novel algorithm outperforms the current state-of-the-art in final performance, convergence rate and robustness to erroneous feedback in OpenAI Gym continuous control benchmarks, both for simulated and real human teachers.

1 INTRODUCTION

In contrast to autonomous Machine Learning techniques, humans are very effective in inferring suitable control strategies when facing new problems. Specifically for intuitive problems, like picking up objects or playing simple games, humans are able to achieve decent performance on first try (Hessel et al., 2018). Communicating this domain knowledge has shown to drastically accelerate model-free control techniques. For example, a well known approach is the Learning from Demonstration (LfD) framework, where a policy is derived using examples of proper execution (Ross et al., 2011). Other methods, like Apprenticeship Learning, employ demonstration to reversely derive the trainer's choices for autonomous improvement (Abbeel and Ng, 2004). To avoid superfluous (and possibly expensive) interaction with the trainer, Gräve and Behnke (2013); Losey and O'Malley (2018) improved sample efficiency with an Active Learning (AL) framework where demonstrations are inquired especially for uncertain policy executions.

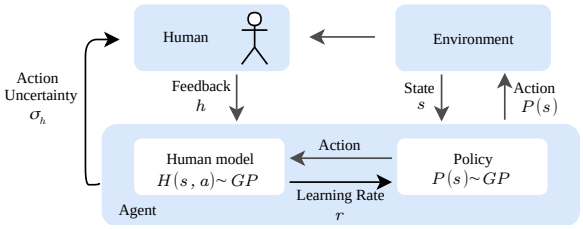


Figure 1: A schematic diagram of the feedback framework of Gaussian Process Coach (GPC). The teacher provides feedback h to corrected the observed action in the respective state.

LfD could however be troublesome for systems that feature agile dynamics. Moreover, the demonstrations require expert knowledge of the system and the solution (Argall et al., 2009). A less demanding approach has been studied by, e.g., Griffith et al. (2013); Knox and Stone (2009), where the trainer gives scalar reward signals (*evaluative* feedback) in response to the agent's observed behavior. Thomaz and Breazeal (2006) however argue that trainers implicitly guide in their reward signal, and base their feedback not solely on past actions but also

* D. Wout, J. Scholten, C. Celemin and J. Kober are with the Department of Cognitive Robotics at the Delft University of Technology, the Netherlands. DAAN090@GMAIL.COM, JAN@JJSCHOLTEN.NL, {C.CELEMIN,J.KOBER}@TUDELFT.NL

on what is going to happen. This intrinsic preference in guidance has been studied by Celemin and Ruiz-del Solar (2015) and resulted in COrrective Advice Communicated by Humans (COACH), an algorithm that allows teachers to shape the optimal policy by providing *corrective* feedback, i.e. in the action domain. This approach engages the users intuition without requiring expertise on the task. Moreover, teachers are now able to guide a policy rather than to evaluate it, which has shown to be better scalable to high-dimensional problems (Suay and Chernova, 2011). COACH has shown to be very efficient on intuitive problems and outperforms evaluative approaches in human assisted learning. However, COACH employs Radial Basis Function (RBF) networks, which require extensive design procedures. The application of COACH is therefore limited to simple and confined problems and hence does not exploit the full potential of corrective feedback implementations.

In order to improve on this point, we introduce GPC, a corrective feedback framework following COACH's structure but comprising engineering advantages by introducing Gaussian Processes (GPs) as alternative to RBF networks. In addition, we employ the available uncertainty estimations of GPs for 1) A pioneering approach of employing AL in an corrective feedback framework, analogue to what Gräve and Behnke (2013); Maeda et al. (2017) do with LfD. 2) Match the learning rate to the learning phase of the teacher and adapt policy corrections accordingly. We apply the novel algorithm to several benchmarks of the OpenAI gym (Brockman et al., 2016), both with simulated and real human teachers, to show the significance of the proposed contributions and the performance in a comparison against previous work.

This study is organized as follows: background material is covered in Section 2. Section 3 details the novel algorithm GPC. The experimental setup and corresponding results are presented in Section 4 and Section 5 respectively.

2 BACKGROUND

In the following, we detail the key components of GPC, starting with COACH which is the basis of the novel framework. The principles of GP are covered thereafter.

2.1 COACH

COrrective Advice Communicated by Humans (COACH), proposed by Celemin and Ruiz-del Solar (2015), is an algorithm that trains agents with corrective advice. It has policy $P_c : S \rightarrow \mathbb{R}^n$, with S the set of states and n the action-space dimensionality, that maps states to continuous actions. The trainer observes the agent and occasionally suggests to either increase or decrease the action. This feedback $h \in \{-1, 1\}$ is modelled in the *human feedback model*: $H_c : S \rightarrow \mathbb{R}^n$. The parameterization of both models is done by RBF networks, where the respective models have different weight to the feature vector $\phi(\mathbf{s}_k)$, with $\mathbf{s}_k$ denoting the state in time-step k. The learning framework is supported by the following modules.

Algorithm 1 COACH framework

Given:
  Policy learning rate e
  Human model learning rate β
  Constant learning rate c_c
  Feature space function $\phi(\cdot)$
for all k **do**
  Get state $\mathbf{s_k}$
  Compute new action $a_k \leftarrow \theta_k \phi(\mathbf{s_k})$
  Obtain corrective advise h
  if $h \neq 0$ **then**
    $H(\mathbf{s_k}) = \psi_k^T \phi(\mathbf{s_k})$
    $\Delta\psi = \beta(h - H(\mathbf{s}_k))\phi(\mathbf{s}_k)$
    Human model update $\psi_{k+1} = \psi_k + \Delta\psi$
    Get learning rate $\alpha(\mathbf{s}_k) = |H(\mathbf{s}_k)| + c_c$
    $\Delta\theta = \alpha(\mathbf{s}_k)\phi(\mathbf{s}_k)he$
    Policy update $\theta_{k+1} = \theta_k + \Delta\theta$
  end if
end for

2.1.1 Policy Supervised Learner

The policy $P_c(\mathbf{s}_k)$ provides the action a for a given state $\mathbf{s}_k$, by taking the linear combination of the weights and the feature vector, i.e. $a_k = P_c(\mathbf{s}_k) = \theta_k^T \phi(\mathbf{s}_k)$, with θ the weight vector of the policy. For every directive correction h given by the teacher, the weight vector is updated according to a Stochastic Gradient Descent approach:

$$\theta_{k+1} = \theta_k - \alpha(\mathbf{s}_k)\nabla_\theta J(\theta),$$

with $\alpha(\mathbf{s}_k)$ the learning rate (obtained as described in Section 2.1.2) and $J(\theta)$ denoting the cost function, which is the squared error between the applied and 'desired' action, given by h and magnitude e. The latter denotes a free parameter set by the user within the range of the action domain. Hence, taking the human feedback into account, the gradient becomes

$$\theta_{k+1} = \theta_k + \alpha(\mathbf{s}_k)\phi(\mathbf{s}_k)he. \quad (1)$$

2.1.2 Human Feedback Supervised Learner

This module models the feedback of the trainer as a function of the state $\mathbf{s}_k$. The predictions are given by a lin-

ear combination of the human model weights ψ_k and the feature vector $\phi(\mathbf{s}_k)$, i.e. $H(\mathbf{s}_k) = \psi_k \phi(\mathbf{s}_k)$. The updates on the weight vector ψ_k are conducted in the same fashion as (1), but now with a known error magnitude $e_h = h - H(\mathbf{s}_k)$ such that

$$\psi_{k+1} = \psi_k + \beta(h - H(\mathbf{s}_k))\phi(\mathbf{s}_k)$$

with β the learning rate of the human model. By this human model, the learning rate in (1) is given as

$$\alpha(\mathbf{s}_k) = |H(\mathbf{s}_k)| + c_c. \tag{2}$$

Note that $H(\mathbf{s}_k) \approx 1$ for consistent feedback with equal sign and hence increases the learning steps. For alternating feedback the learning rate diminishes. To prevent the learning rate from dwindling to zero, (2) is appended with a constant factor c_c.

The outline of the COACH framework is depicted in Algorithm 1. Lines 3-5 comprise of policy executions. The update lines consist of the human prediction and human model updates (line 7-10) as motivated in Section 2.1.2. The policy updates from Section 2.1.1 are subsequently given in lines 11-12. Furthermore, the COACH framework can be extended by the *Credit Assigner*, which takes a human delay into account for the feedback given by the teacher. Since this study will not exploit this feature it will not be covered here.

2.2 GAUSSIAN PROCESSES

Gaussian Processes (GPs) are Bayesian non-parametric function approximation models. It is a collection of random variables, such that every finite collection of those random variables has a multivariate normal distribution. GPs do not require specification of a model structure a priori and provide the uncertainty along with the predictions. A GP is fully specified by its mean $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, i.e.

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

Let $\mathbf{y} = \{y_1, ...y_n\}$ be a set of observations from a stochastic process

$$y_i = f(\mathbf{x}_i) + \epsilon, \tag{3}$$

where $\mathbf{x}_i$ denotes the input vector of observation y_i. The noise ϵ is assumed Gaussian with standard deviation σ_o. The input matrix is defined as $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$. Applying the conditional distributions (Rasmussen and Williams, 2006), the following posterior predictive equations for test inputs $\mathbf{x}_*$ are given as:

$$\begin{aligned}
\mathbf{f}_*|X, \mathbf{y}, \mathbf{x}_* &\sim \mathcal{N}(\bar{\mathbf{f}}_*, \mathrm{cov}(\mathbf{f}_*)), \quad \text{where} \\
\bar{\mathbf{f}}_* &= m(\mathbf{x}_*) + K_*[K + \sigma_n^2 I]^{-1}(\mathbf{y} - m(X)), \\
\mathrm{cov}(\mathbf{f}_*) &= K_{**} - K_*[K + \sigma_n^2 I]^{-1} K_*,
\end{aligned}$$

where $K_* = k(X, \mathbf{x}_*)$, $K_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$, and K is the Gram matrix with entries $K_{ij} = k(x_i, x_j)$. The Gaussian noise per observation is denoted as σ_n and has a similar function as ϵ in (3). The kernel function $k(\mathbf{x}, \mathbf{x}')$ is a measure of similarity between two input vectors $\mathbf{x}$ and $\mathbf{x}'$. In this study, we employ two kernel functions. The first one is the squared exponential (SE) kernel, which is given as

$$k_s(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2l^2}\right), \tag{4}$$

with $\beta_r = \{\sigma_s, l\}$ the *hyperparameters* of the kernel function, consisting of the signal variance σ_s and length-scale l. The length-scale denotes a measure for the roughness of the data. In general, one can assume that extrapolating more than l units away from the input data is considered unreliable. The second kernel function, the Matérn kernel, is specified as

$$k_m(\mathbf{x}, \mathbf{x}') = \sigma_m^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{2}{l}\right)^{\nu} B_v \left(\sqrt{2\nu}\frac{|\mathbf{x} - \mathbf{x}'|}{l}\right) \tag{5}$$

with $B_v(\cdot)$ the modified Bessel function (Abramowitz and Stegun, 1965), $\Gamma(\cdot)$ the Gamma function and the *hyperparameters* $\beta_m = \{\sigma_m, l, \nu\}$. Here, ν denotes a 'smoothness' parameter that correlates with the amount of times the target function is differentiable (Rasmussen and Williams, 2006).

For multivariate targets, we train conditionally independent GPs for each target dimension.

3 GAUSSIAN PROCESS COACH (GPC)

We now introduce Gaussian Process Coach (GPC), an algorithm based on COACH that employs GP as an alternative to RBF networks to comprise advantage in scaling and sample efficiency. A schematic of the method is depicted in Fig. 1. In the main format, the trainer observes the environment and the current policy and provides action corrections to advance the policy. These corrections trigger agent updates in order to take immediate effect on the policy. This process is repeated until convergence. The pseudo-code of GPC is in Algorithm 2.

This section defines GPC for a one dimensional action-space, but scales straightforwardly to higher dimensional problems.

3.1 MODELLING POLICY AND FEEDBACK

The GPC framework engages two GP models: the policy P and the human model H. The prior of the policy is modelled as:

$$P : S \to \mathbb{R} \sim \mathcal{GP}(m_p(\mathbf{s}), k_p(\mathbf{s}, \mathbf{s}')), \tag{6}$$

Algorithm 2 GPC Algorithm

1: **Given:**
   Kernels $k(\cdot)$ for H and P
   Hyperparameters β with M_{cs} or M_{ns}
   Constant learning rate c_r
2: **for** all k within episode **do**
3: Get state $\mathbf{s_k}$
4: Execute action $a_k = P(\mathbf{s}_k)$ and obtain $\sigma_p(\mathbf{s}_k)$
5: Obtain corrective advise $h \in \{-1, 1\}$
6: **if** $h \neq 0$ **then**
7: $\mathbf{z_k} = (\mathbf{s}_k, a_k)$
8: $h_{\text{est}} = H(\mathbf{z}_k)$ with uncertainty $\sigma_h(\mathbf{z}_k)$
9: Learning rate $r_k \leftarrow \sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r$
10: New action $a_n = a_k + r_k \cdot h_k$
11: Update dictionary P and apply SPARS(N_p)
12: Update dictionary H: $N_h = \{..., (\mathbf{z}_k, h_k)\}$
13: Update $M_p, M_h \leftarrow \text{cov}(N_p, N_h)$ // NS only
14: Train GPs: TRAIN(P, H)
15: **end if**
16: **end for**

Here, $m_p(\mathbf{s})$ is assumed 0. The policy is trained with the set $N_p = \{(\mathbf{s}_1, a_1), (\mathbf{s}_2, a_2), ..., (\mathbf{s}_m, a_m)\}$, which contains state-action data derived from the directional feedback from the trainer (details in Section 3.3.3). The human feedback is modelled by

$$H : S \times A \to \mathbb{R} \sim \mathcal{GP}(m_h(\mathbf{z}), k_h(\mathbf{z}, \mathbf{z}')), \quad (7)$$

with A the action space. The mean is assumed $m_h(\mathbf{z}) = 0$. This human model is trained with the set $N_h = \{(\mathbf{z}_1, h_1), (\mathbf{z}_2, h_2), ..., (\mathbf{z}_v, h_v)\}$, where $\mathbf{z}$ denote the concatenation of state $\mathbf{s}$ and action a, and $h \in \{-1, 1\}$ the suggested action correction of the teacher (decrease or increase). The proposed GPC introduces a different human model with respect to the one of COACH, where the feedback was only state dependent, i.e. $H_c : S \to \mathbb{R}$ (see Section 2.1). We have integrated the action in our human model to infer the human feedback per state-action, rather than state only. Further details on this principle are provided in Section 3.3, where we elaborate on the uncertainty advantages.

Both models of GPC require a kernel function that represents how the target function and uncertainty propagates along the input dimensions. For the human model H we assume a smooth propagation of the target function and therefore adopt the SE kernel (4). To allow for more freedom in the policy function in terms of roughness and discontinuities, we adopt the Matérn kernel (5) for P (Duvenaud, 2014; Rasmussen and Williams, 2006).

3.2 FEATURE SCALING

The policy in (6) and human model in (7) both concern a multidimensional regression on the input data. Each input dimension may however be subject to data with completely different orders of magnitude, such that a single length-scale is unsuitable. We therefore take an approach that allows us to set an independent length-scale per input dimension.

Let us consider the SE kernel from (4). Following Rasmussen and Williams (2006), the parameterization in terms of the *hyperparameters* results in

$$k_s(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T l M (\mathbf{x} - \mathbf{x}')\right),$$

with M the diagonal matrix consisting of the *characteristic length-scales* per axis. Such a covariance function implements Automatic Relevance Determination (ARD) (Neal, 1995). This study adopts two distinct methods for determining the diagonal values of M. In the first approach we let the trainer decide on the respective relevance of the input dimensions:

$$M_{cs} = \text{diag}(\mathbf{w})^{-2},$$

with $\mathbf{w}$ a vector consisting of custom 'weights' on the input dimensions. These values are determined a priori and deemed static throughout the learning process. This method is referred to as GPC(-CS). The second method concerns the normalization of the independent inputs for an equal relative dependency, resulting in an approach where any length-scale tuning is circumvented. The result is an approach that does not scale with the input dimension and could hence be decisive for higher-order systems. The parameterization is carried out by

$$M_{ns} = \text{diag}(\sigma_m)^{-2},$$

with σ_m the vector containing the variance of the independent input dimensions, which is updated for every feedback sample (see line 13 in Algorithm 2). This method will be referred to as GPC-NS.

The extension to the Matérn kernel (5) is straightforward with $M_{cs} = \text{diag}(\mathbf{w})$ and $M_{ns} = \text{diag}(\sigma_m)$ for every length-scale l. To distinguish between the scaling of the policy and the human model we add subscript h and p, e.g. $M_{cs,h}$.

3.3 LEVERAGING UNCERTAINTY

GPs provide uncertainty estimates with every query point based on dissimilarity with the training data. For the policy, the uncertainty reflects the presence of feedback data in the respective or surrounding state. Due to the integration of the action in the input of the human model, this

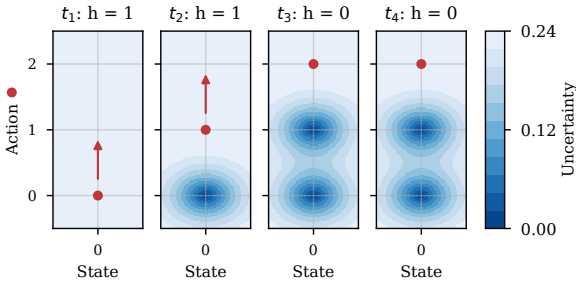


Figure 2: This hypothetical situation clarifies how the action and uncertainty evolves with every feedback sample. The uncertainty is shown for the human model H and is reduced for a (s, a)-pair as feedback is obtained.

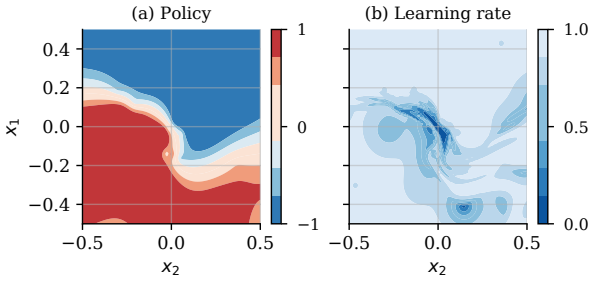


Figure 3: Snapshot of a policy *(a)* and learning rate *(b)* for controlling a system around an unstable equilibrium $(0, 0)$.

uncertainty reflects the presence of feedback for state-action pairs. To elaborate on this, a hypothetical example is depicted in Fig. 2. The contiguous plots show the evolution of the policy and its uncertainty as new feedback is obtained. We may envision this principle as building a map that discloses certain and uncertain regions with respect to past feedback. This feature comprises the main advantage of GPC over other methods.

3.3.1 Adaptive Learning Rate (ALR)

We assume that the teacher encounters two teaching phases during the learning period. The initial learning phase arises when the process is commenced and the policy is idle. We believe that the feedback in this stage will mostly concern raw adjustments in order to create a coarse version of the final policy. These coarse adaptations will gradually shift towards the second learning phase where trainers apply small refinements to the policy for meticulous improvements. In this study, we model the transition from coarse to fine adjustments not as a universal annealing process. Instead, we adapt the learning rate to the intended correction per state.

Hence, we introduce the following Adaptive Learning Rate (ALR):

$$r_k = \sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r, \tag{8}$$

with r_k the learning rate, $\mathbf{s}_k$ the state and $\mathbf{z}_k$ the concatenation of $(\mathbf{s}_k, a_k)$ (see line 7-9 in Algorithm 2). The uncertainty of the policy σ_p allows us to accelerate the learning by increasing the learning rate for the first feedback instances. The uncertainty estimation of σ_h adopts a high value for consistent feedback (see Fig. 2). As soon as alternating feedback is given, the uncertainty, and thus the learning rate decreases to allow for refinements. The parameter c_r denotes the constant rate and prevents stagnation in the event that $\sigma_{p,h} \approx 0$. GPC differs from COACH for updating the policy, since the error magnitude e is now implicitly included in the computation of r_k in (8).

An example of the policy and the learning rate during a learning process is depicted in Fig. 3. It shows an environment with a two-dimensional continuous state-space and an unstable equilibrium as reference at $(x_1, x_2) = (0, 0)$. The policy *(a)* is trained by a teacher employing the ALR. The corresponding learning rate is displayed in *(b)*. Note that for critical states (area around $(x_1, x_2) = (0, 0)$) alternating feedback has caused the ALR to decrease, such that the policy can be refined.

3.3.2 Active Learning (AL)

The available uncertainty of the GPs can be used in an AL framework (Chernova and Thomaz, 2014), where high-informative feedback can be inquired for uncertain actions. Recent studies have shown great performance improvements with agent-induced feedback, mostly in the LfD domain (Gräve and Behnke, 2013; Losey and O'Malley, 2018). This study is, to the authors' knowledge, the first to assess the potential with a directional feedback framework. Other than Chernova and Veloso (2009), who employed AL with the uncertainty of the visited state, we believe that especially the uncertainty in the action can advance the convergence of directive feedback methods. The motivation for this reasoning is that, in contrast to the LfD paradigm, inquiring human assistance in terms of feedback does not yield the optimal action instantaneously. In GPC - and feedback implementations in general - multiple feedback instances are needed to approach the optimal policy. Hence, rather than employing state uncertainty, we apply uncertainty per action, which is obtained by the human model

$$\Delta_k = c_a \sigma_h(\mathbf{z}_k), \tag{9}$$

with $\mathbf{z}_k$ the same as in (7) and gain c_a to decouple AL from the ALR (see (8)). By inquiring feedback for high values of Δ_k we prioritize consistent feedback, since inconsistent feedback would reduce Δ_k. AL will therefore

Algorithm 3 Sparsification of policy P training data

1: **function** SPARS($\mathbf{s}_k$, a_n, σ_p, σ_{thres}, N_p)
2: **if** $\sigma_p < \sigma_{\text{thres}}$ **then**
3: index $\leftarrow \arg\max_i \; k_p(\mathbf{s}_i, \mathbf{s}_k)$
4: $N_p(\text{index}) \leftarrow (\mathbf{s}_k, a_n)$
5: **else**
6: Append dictionary $N_p = \{\ldots, (\mathbf{s}_k, a_n)\}$
7: **end if**
8: **return** N_p
9: **end function**

further aid in establishing an inaccurate but rather complete policy as early as possible, before proceeding to the refinement stage.

3.3.3 Sparsification for Corrective Learning

For every feedback instance provided by the trainer, the dictionary of the policy P is appended with the new tuple:

$$N_p = \{\ldots, (\mathbf{s}_{m+1}, a_{m+1})\} \tag{10}$$

where $(\mathbf{s}_{m+1}, a_{m+1})$ is calculated based on the executed action a_k, learning rate r_k and feedback h_k, i.e.

$$a_{m+1} = a_k + r_k h_k.$$

This approach renders the previous action a_k obsolete. In this application, a deficient property of GPs that hinders convergence is that by appending the dictionary following (10), the updated action on $\mathbf{s}_{m+1}$ is an average of a_k and a_{m+1} (assuming a coinciding or adjoining data instance). We therefore propose a sparsification method in which the tuple most relevant to the obsolete action a_k is omitted, rendering a_{m+1} the new action. Taking relevancy into account while preserving the uncertainty estimations was not found in conventional online sparsification methods (e.g. Nguyen-Tuong and Peters, 2010). We therefore introduce a new sparsification technique that specifically applies to applications with iterative updates on the GP policy model.

The main outline of this sparsification is as follows: for every new feedback instance $(\mathbf{s}_{m+1}, a_{m+1})$, the uncertainty of the policy $\sigma_p(\mathbf{s}_k)$ is compared against a certain threshold σ_{thres}. We set this threshold to

$$\sigma_{\text{thres}} = \frac{1}{2}\sqrt{\sigma^2_{s,m}},$$

with $\sigma^2_{s,m}$ either from (4) or (5). In the event that this threshold is exceeded, the dictionary sample with the biggest covariance (i.e. smallest *Mahalanobis* distance (Mahalanobis, 1936)) is omitted. We thereby prevent the policy from being negatively influenced by obsolete (old) training data. The sparsification method is presented in Algorithm 3 and executed simultaneously with appending N_p, see line 11 in Algorithm 2. The existing input elements of the policy dictionary N_p are denoted by $\mathbf{s}_i$.

Figure 4: A snapshot of each domain used for the experiments. the most left benchmark denotes the Underactuated Inverted Pendulum(-v0), The Cart-Pole(-v0) environment in the middle, and most right the Lunar Lander(-v2) (Brockman et al., 2016). The environments are sorted with respect to complexity.

4 EXPERIMENTAL SETUP

In this section we detail the experiments in which the performance of GPC is evaluated. The tests are carried out in three standardized benchmark problems from the OpenAI Gym (Brockman et al., 2016), namely the Inverted Pendulum(-v0), the Cart-Pole(-v0) and the Lunar Lander(-v2). The experiments with oracles (synthesized feedback source) are introduced to test the performance with consistency for all algorithms. The oracle tests also comprise the AL and ALR assessment. The applicability to the interactive domain is tested in separate experiments with actual human feedback. The performance of GPC[1] will be tested against baseline COACH throughout (Celemin and Ruiz-del Solar, 2015)[2].

4.1 ORACLE TESTS

An oracle simulates human feedback based on a comparison of the executed action with a reference policy. The oracle experiments are carried out to exclude human factors such as inconsistency and limited attention that hinder a fair comparison between methods. Furthermore, it allows to accurately study the robustness of the algorithms to erroneous feedback.

4.1.1 Performance and Robustness tests

First, we will assess the performance of GPC for perfect and erroneous feedback. For the experiments we set a static feedback rate $\gamma = 5\%$. When the action is close to the target action within the range δ, i.e. $|a_k - a^*| = \delta$, the policy is considered converged and receives no more feedback. The robustness of the algorithms is tested by erroneous feedback. In this study, we will adopt error

[1] github.com/DWout/GPC
[2] github.com/rperezdattari/COACH-gym

rates of 10% and 20%, which will be administered following the protocol of Celemin et al. (2018).

4.1.2 Active Learning (AL)

The potential regarding AL is assessed by encouraging feedback for uncertainty policy actions. In order to exclude any random human factors, the performance is measured using an oracle. As such, we will adapt the feedback rate by incorporating the uncertainty of the human model H, i.e.: $\gamma = \Delta_k + \gamma_c$, with Δ_k as in (9) and γ_c denoting the minimum feedback rate. The application of AL is measured against a baseline with the same episodic feedback rate, but not uncertainty triggered (see Table 1, *ii* and *iv*). The ALR is excluded from the test to circumvent any influences. To account for feedback inconsistencies the erroneous feedback likelihood is set to 10%.

4.1.3 Ablation Study

The ablation assessment will analyze the contribution of the ALR (Section 3.3.1). We will run oracle tests employing the learning rate in (8) and compare this to the baseline test from Section 4.1.2 with the same episodic feedback rate and erroneous feedback likelihood. In addition, we will test a combination of AL and ALR. A summary of the experimental setup is presented in Table 1.

Table 1: Overview of the experiments regarding Active Learning (AL) and the ablation study for the Adaptive Learning Rate (ALR). For fair comparison the experiments are conducted with the same feedback (Fb) rate γ.

	AL:	ALR:	Fb rate γ:	Learning rate r_k:
i	✓	✓	$\Delta_k + \gamma_c$	$\sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r$
ii	✓	×	$\Delta_k + \gamma_c$	r_c
iii	×	✓	$\gamma_{\text{ep.avg}}(i)$	$\sigma_p(\mathbf{s}_k) + \sigma_h(\mathbf{z}_k) + c_r$
iv	×	×	$\gamma_{\text{ep.avg}}(ii)$	r_c

4.2 HUMAN TEACHERS

This section will elaborate the experiments for validating the application of GPC to interactive settings. The experiments are conducted by employing four human teachers (in the age of 20 to 30, of different background) to the three proposed benchmarks with the objective to achieve convergence as fast as possible. The participants perform the training with every algorithm for every environment four times: two dummy runs to get acquainted with the environment, and two real runs that are recorded. The tests runs are performed single blind: the participants are not informed about which algorithm they controlled.

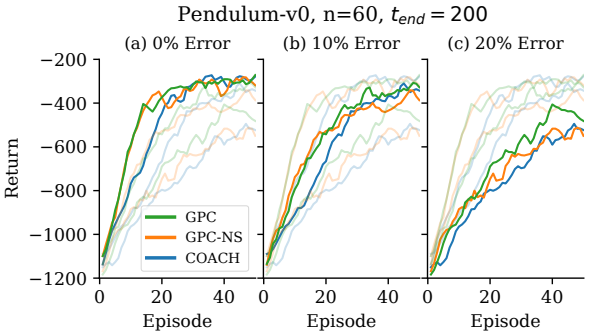


Figure 5: Average environmental return per episode for the Pendulum-v0 domain. GPC and GPC-NS show nearly the same performance, and outperform COACH mainly in the leading domain. The final performance is similar. The performance for every error rate is reprinted (shaded) in all figures for cross-comparison.

5 EXPERIMENTAL RESULTS

In this section, we report GPC's performance on the three domains (see Section 4). The kernels and *hyperparameters* for the human model and the policy are depicted in Table 2. For readability purposes the presented results are a walking mean of 3 samples, unless otherwise specified.

Table 2: Hyperparameters of the GPs in the benchmarks. The policy and human model are modelled by Squared Exponential (SE) and Matérn (Mat.) kernel. The constant learning rate in (8) is denoted as c_r.

	Pendulum		Cart-Pole		Lunar Lander	
	CS	NS	CS	NS	CS	NS
H:	*SE*	*SE*	*SE*	*SE*	*SE*	*SE*
c_h	0.7	0.45	0.01	0.08	0.01	0.08
l_h	0.1	0.1	0.2	0.5	0.2	0.2
P:	*Mat.*	*Mat.*	*Mat.*	*Mat.*	*Mat.*	*Mat.*
c_p	0.01	0.03	0.01	10^{-3}	0.01	10^{-3}
l_p	0.7	0.5	0.2	0.7	0.4	0.6
ν_p	0.5	1.5	1.5	1.5	1.5	1.5
c_r	0.01	0.02	0.02	0.05	0.02	0.05

5.1 PERFORMANCE AND ROBUSTNESS

The return for the Pendulum domain is depicted in Fig. 5. The GPC variants show similar convergence and robustness properties. Due to the coarser exploration in the initial learning phase the learning curve is steeper in comparison to COACH. The final performance is similar. The average learning rate for all consecutive feedback instances is depicted in Fig. 6 for an error rate of 0% and 20%. For GPC we see a more aggressive learning rate for the initial learning phase, which diminishes upon

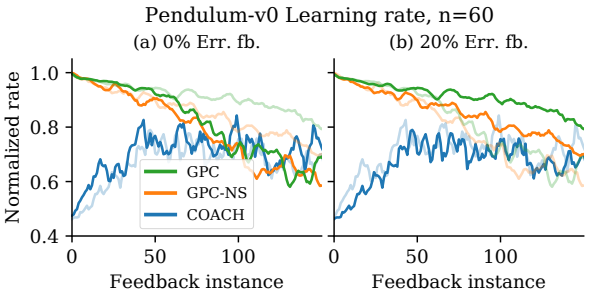


Figure 6: Normalized average learning rate in the Pendulum-v0 domain for both GPC and COACH. In contrast to existing methods, the learning rates of the GPC implementations diminish over time, such that the corrections become more subtle upon convergence. As desired, this reduction develops more gradually in case of erroneous feedback.

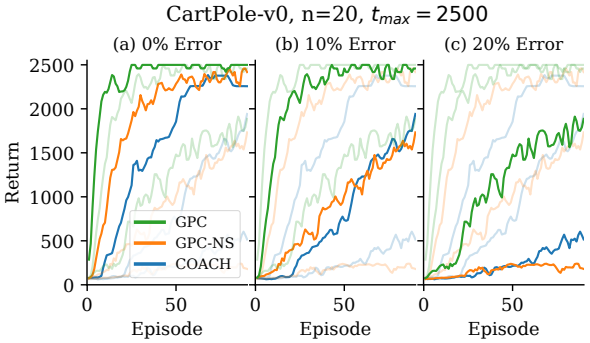


Figure 7: Average return per episode for the CartPole-v0 domain. Both GPC implementations outperform COACH for ideal feedback. GPC shows good robustness to erroneous feedback, whereas GPC-NS is more brittle.

convergence. For 20% erroneous feedback this propagation is more gradual, as it should be to reflect on the impeded learning where refinements are appropriate at a later time.

The steeper initial learning curve, which was observed in the Pendulum domain in Fig. 5, also distinguishes GPC from COACH in the Cart-Pole environment (see Fig. 7). The protracted take-off time for COACH is presumably a result of the *human feedback supervised learner* module (see Celemin and Ruiz-del Solar, 2015) that adopts a reduced learning rate for the initial learning phase. For every error rate GPC outperforms COACH. The tuning convenience for GPC-NS shows to trade with some suboptimal performance for erroneous feedback.

The performance of GPC and COACH in the Lunar Lander domain is depicted in Fig. 8. Both GPC and GPC-NS outperform COACH for every error rate. COACH was

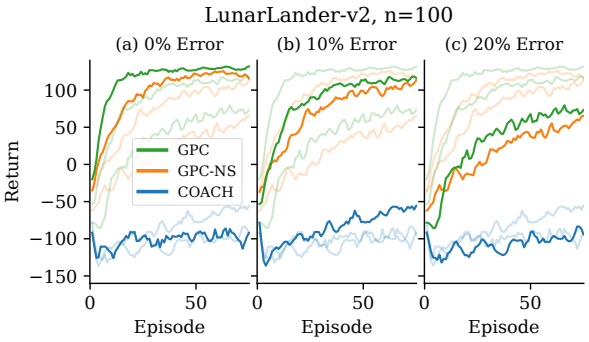


Figure 8: Average return for the LunarLander-v2 environment. Both GPC implementations achieve good performance either for ideal or erroneous feedback. COACH yields poor performance due to intractability of the feature space, which is custom parameterized in $\mathbb{R}^{24}$.

unable to achieve any performance due to the unfeasible manual parameterization of the feature space in $\mathbb{R}^{24}$ (lower/upper bound and interval for every input dimension (Busoniu et al., 2010)).

5.2 ACTIVE LEARNING AND ABLATION

The result of the four test cases from Table 1 are displayed in Fig. 9 for the Cart-Pole environment with constant feedback likelihood of $\gamma_c = 0.01$, constant additive learning rate of $c_r = 0.01$ and a static learning rate of $r_c = 0.4$. AL combined with ALR has superior performance. The individual components (ALR and AL resp.) both prove their significance compared to the baseline. The average learning rate for the ALR tests measures 0.0386 for i and 0.0374 for iii, which is lower on average but better balanced to the static rate of $r_c = 0.4$ in ii and iv.

5.3 HUMAN VALIDATION

The performance for all environments is depicted in Fig. 10. For the Inverted Pendulum and the Cart-Pole environment both COACH and GPC converge to maximal performance. Although some relative differences are noticeable in the learning curve, the variations are not deemed statistically significant considering the number of testst. The fact that the lacking robustness of the COACH implementation (Fig. 7) does not emerge in this result is notable. In contrast to the static behavior of oracles, humans anticipate to the consequences of the provided feedback and adapt their feedback strategy accordingly. When the corrections at a particular state are deemed insufficient, teachers may choose to provide multiple feedback samples subsequently in order to realize the intended effect. An interesting observation is

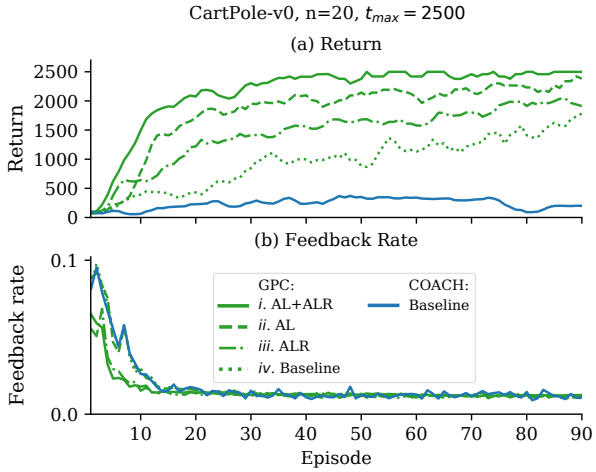


Figure 9: Average return *(a)* and feedback rate *(b)* for the Active Learning (AL) and the ablation of Adaptive Learning Rate (ALR) in the Cart-Pole domain. AL and ALR combined achieve superior performance. It shows that the Adaptive Learning Rate accelerates the convergence with less feedback.

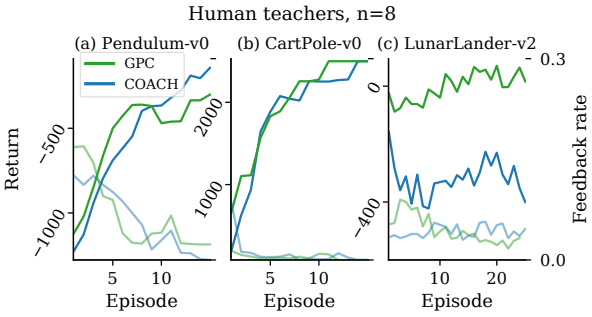


Figure 10: Average return of eight experiments from four human teachers on the three adopted domains. Performance is similar to oracle tests and validate the suitability of GPC to the interactive domain.

the difference in return for the Lunar Lander environment in *(c)*, which validate the findings from the oracle benchmarks in Fig. 8: The unfeasible parameterization of the feature space severely deteriorate the performance in higher dimensional problems.

6 CONCLUSION

Humans are very efficient in understanding control strategies using intuition and common sense. Corrective feedback is an especially effective means of communication and the current state-of-the-art, COrrective Advice Communicated by Humans (COACH), enables one to establish a control law without requiring control or engineering expertise. Moreover, performance is superior over methods that learn autonomously or from evaluative feedback. However, COACH employs Radial Basis Function (RBF) networks for modelling which requires meticulous feature space engineering before these advantages enter into force.

In this work, we have presented GPC. It has an architecture similar to COACH, but it engages Gaussian Processes (GPs) such that modelling expertise is no longer required and the limitation to confined problems is hereby overcome. Moreover, we leverage the available uncertainty with an Adaptive Learning Rate (ALR) that adapts to the trainer's learning phase. In addition, we introduced a new sparsification technique, specifically designed for efficient and accelerated GP policy updates. GPC was applied to three continuous benchmarks from the OpenAI Gym: the Inverted Pendulum, Cart Pole, and Lunar Lander. Our novel framework outperform COACH on every domain tested by means of faster learning and better robustness to erroneous feedback. The greatest improvement was for the Lunar Lander problem, where RBF parameterization fails but GPC is flawless.

In addition to the performance and robustness assessment, we performed two additional studies: 1) We have addressed the potential of Active Learning (AL) and demonstrated how eliciting feedback for actions with greatest uncertainty yields drastic improvements on convergence. 2) We have furthermore presented an alternative implementation GPC-NS where length-scale tuning is circumvented by online normalization of the input space. It is slightly suboptimal and trades some robustness in comparison to GPC, but a great advantage is that it does not require any input parameterization in new domains. This could especially be decisive in higher-dimensional problems, and furthermore renders our work feasible also for non-experts.

In future work, it might be possible to further innovate the dynamical scaling, such that the applicability and generality of GPC-NS is again extended. In addition, the AL opportunities assessed here deserve further research and should be validated also with human participants.

References

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Abramowitz, M. and Stegun, I. A. (1965). *Handbook of Mathematical Functions: With Formulas, Graphs,*

and Mathematical Tables, volume 55. Courier Corporation.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym. arXiv:1606.01540 [cs.LG].

Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming using Function Approximators*. CRC press.

Celemin, C. and Ruiz-del Solar, J. (2015). COACH: Learning continuous actions from corrective advice communicated by humans. In *IEEE International Conference on Advanced Robotics (ICAR)*, pages 581–586.

Celemin, C., Ruiz-del Solar, J., and Kober, J. (2018). A fast hybrid reinforcement learning framework with human corrective feedback. *Autonomous Robots*, First Online.

Chernova, S. and Thomaz, A. L. (2014). Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121.

Chernova, S. and Veloso, M. (2009). Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25.

Duvenaud, D. (2014). *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge.

Gräve, K. and Behnke, S. (2013). Learning sequential tasks interactively from demonstrations and own experience. In *IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)*, pages 3237–3243.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. L. (2013). Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2625–2633.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Knox, W. B. and Stone, P. (2009). Interactively shaping agents via human reinforcement: The TAMER framework. In *International Conference on Knowledge Capture*, pages 9–16.

Losey, D. P. and O'Malley, M. K. (2018). Including uncertainty when learning from human corrections. arXiv:1806.02454 [cs.RO].

Maeda, G., Ewerton, M., Osa, T., Busch, B., and Peters, J. (2017). Active incremental learning of robot movement primitives. In *Annual Conference on Robot Learning (CoRL)*, pages 37–46.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. volume 2, pages 49–55. National Institute of Science of India.

Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

Nguyen-Tuong, D. and Peters, J. (2010). Incremental sparsification for real-time online model learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 557–564.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*, volume 2. MIT Press Cambridge, MA.

Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635.

Suay, H. B. and Chernova, S. (2011). Effect of human guidance and state space size on interactive reinforcement learning. In *IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*.

Thomaz, A. L. and Breazeal, C. (2006). Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 6, pages 1000–1005.

Bibliography

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM.

Abramowitz, M. and Stegun, I. A. (1965). *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation.

Argall, B. D. (2009). Learning mobile robot motion control from demonstration and corrective feedback. *Diss. University of Southern California*.

Argall, B. D., Browning, B., and Veloso, M. (2008). Learning robot motion control with demonstration and advice-operators. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 399–404. IEEE.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Celemin, C. and Ruiz-del Solar, J. (2015). Coach: Learning continuous actions from corrective advice communicated by humans. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 581–586. IEEE.

Celemin Paez, C., Ruiz-del Solar, J., and Kober, J. (2018). A fast hybrid reinforcement learning framework with human corrective feedback. *Autonomous Robots*, 0(99).

Chernova, S. and Thomaz, A. L. (2014). Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121.

Chernova, S. and Veloso, M. (2009). Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25.

Csató, L. and Opper, M. (2002). Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668.

Duvenaud, D. (2014). *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge.

Gräve, K. and Behnke, S. (2013). Learning sequential tasks interactively from demonstrations and own experience. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3237–3243. IEEE.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. L. (2013). Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*, pages 2625–2633.

Heess, N., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, A., Riedmiller, M., et al. (2017). Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Knox, W. B. and Stone, P. (2009). Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Kober, J. and Peters, J. (2009). Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*, pages 2112–2118. IEEE.

Kober, J. and Peters, J. (2012). Reinforcement learning in robotics: A survey. In *Reinforcement Learning*, pages 579–610. Springer.

Le, Q. V., Smola, A. J., and Canu, S. (2005). Heteroscedastic gaussian process regression. In *Proceedings of the 22nd international conference on Machine learning*, pages 489–496. ACM.

Li, G., Whiteson, S., Knox, W. B., and Hung, H. (2016). Using informative behavior to increase engagement while learning from human reward. *Autonomous agents and multi-agent systems*, 30(5):826–848.

Loftin, R. T., MacGlashan, J., Peng, B., Taylor, M. E., Littman, M. L., Huang, J., and Roberts, D. L. (2014). A strategy-aware technique for learning behaviors from discrete human feedback. In *AAAI*, pages 937–943.

Losey, D. P. and O'Malley, M. K. (2018). Including uncertainty when learning from human corrections. *CoRR*, abs/1806.02454.

Maeda, G., Ewerton, M., Osa, T., Busch, B., and Peters, J. (2017). Active incremental learning of robot movement primitives. In *CoRL 2017-1st Annual Conference on Robot Learning*, pages 37–46.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. volume 2, pages 49—55. National Institute of Science of India.

Meriçli, Ç., Veloso, M., and Akın, H. L. (2011). Task refinement for autonomous robots using complementary corrective human feedback. *International Journal of Advanced Robotic Systems*, 8(2):16.

Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Ng, A. Y., Russell, S. J., et al. (2000). Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670.

Nguyen-Tuong, D. and Peters, J. (2010). Incremental sparsification for real-time online model learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 557–564.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*, volume 2. MIT Press Cambridge, MA.

Shin, M. and Goel, A. L. (2000). Empirical data modeling in software engineering using radial basis functions. *IEEE Transactions on Software Engineering*, 26(6):567–576.

Suay, H. B. and Chernova, S. (2011). Effect of human guidance and state space size on interactive reinforcement learning. In *2011 RO-MAN*, pages 1–6.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 2 edition.

Thomaz, A. L., Breazeal, C., et al. (2006). Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Aaai*, volume 6, pages 1000–1005. Boston, MA.

Vien, N. A. and Ertel, W. (2012). Reinforcement learning combined with human feedback in continuous state and action spaces. In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6. IEEE.

Vollmer, A.-L. and Hemion, N. J. (2017). Robot skill learning with user feedback: evaluating system performance and human factors. *Preprint: http://hemion.org/n/*.

Wahba, G. (1990). *Spline models for observational data*, volume 59. Siam.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College, Cambridge.

Glossary

AL Active Learning

ALR Adaptive Learning Rate

ARD Automatic Relevance Determination

COACH COrrective Advice Communicated by Humans

GP Gaussian Process

GPC Gaussian Process Coach

LfD Learning from Demonstration

MDP Markov Decision Process

RL Reinforcement Learning

RBF Radial Basis Function

SE Squared Exponential