Deep Reinforcement Learning

with

Feedback-based Exploration

J.J. Scholten

Master Thesis
Systems & Control
March 29th, 2019

TU Delft

Deep Reinforcement Learning

with Feedback-based Exploration

by

J.J. Scholten

to obtain the degree of Master of Science at the Delft University of Technology,
to be defended publicly on Friday March 29, 2019 at 13:30.

Student number:	4253493	
Project duration:	February 2018 - March 2019	
Thesis committee:	dr. ir. J. de Winter,	chair
	dr. ing. J. Kober,	supervisor
	dr. C. Celemin Paez,	supervisor
	dr. S. Wahls	

Abstract

Deep Reinforcement Learning enables us to control increasingly complex and high-dimensional problems. Modelling and control design is longer required, which paves the way to numerous innovations, such as optimal control of evermore sophisticated robotic systems, fast and efficient scheduling and logistics, effective personal drug dosing schemes that minimise complications, as well as applications not yet conceived. Yet, this potential is obstructed by the need for vast amounts of data. Without it, deep Reinforcement Learning (RL) cannot work. If we want to advance RL research and its applications, a primary concern is to improve this sample efficiency. Otherwise, all potential is restricted to settings where interaction is abundant, whilst this is seldom the case in real-world scenarios.

In this thesis we will study binary corrective feedback as a general and intuitive manner to incorporate human intuition and domain knowledge in model-free machine learning. In accordance with our conclusions drawn from literature, we will present two algorithms, namely Probabilistic Merging of Policies (PMP) and its extension Predictive PMP (PPMP). Both methods estimate the abilities of their inbuilt Reinforcement Learning (RL) entity by computing the covariance over multiple output heads of the actor network. Subsequently, the corrections are quantified by comparing the uncertainty in what is learned with the inaccuracy of the given feedback. The resulting new action estimates will immediately be applied as probabilistic conditional exploration. The first algorithm is a surprisingly clean and straightforward way to accelerate an off-policy RL baseline and as well improves on existing work that learns from corrections only. Its extension Predictive Probabilistic Merging of Policies (PPMP) predicts the corrected samples. This gives the most substantial improvements, whilst the required feedback is further reduced.

We demonstrate our algorithms in combination with Deep Deterministic Policy Gradient (DDPG) on continuous control problems of the OpenAI Gym. We show that the greatest part of the otherwise ignorant learning process is indeed evaded. Moreover, we achieve drastic improvements in final performance, robustness to erroneous feedback and feedback efficiency both for simulated and real human feedback, and show that our method is able to outperform the demonstrator.

“He who learns but does not think, is lost! He who thinks but does not learn is in great danger.”

— *Confucius*

Preface

Delft, January 2019

In the year 1913, when Giacomo Balla painted the artwork "Automobile in Corsa", that covers this thesis, his paint captured the spirit of the time. It was the same year in which the first mass-produced car, the T-Ford, came into being. Many were astonished by the revolutionary innovations that were rapidly changing society, and the repeated fragmented evocations of the car's speed illustrate how the new technologies had an overwhelming effect on those who experienced them for the first time.

Years later, the industrial revolution had disrupted the labour markets, there had been wars, and the public faith in technology had turned down; it was no longer deemed solely a driving factor of human progress. Sir Charles Spencer Chaplin phrased this disenchantment best: *"We have developed speed, but we have shut ourselves in. Machinery that gives abundance has left us in want"*. If we again observe the painting, which illustrates the notion of speed, innovation, perplexity, and the corresponding excitement so well, it occurs to me that these emotions are again broadly felt for the current developments in data, learning, and artificial intelligence — the 'digital revolution'.

With this work, I have developed my understanding of artificial intelligence. I have thereby as well placed myself in the midst of contemporary discussion on what technology can bring us, and what it could take. Concerns about privacy, labour, surveillance, the global monopolies... there are many plausible directions in which machine learning can lead to disappointment, if we do not scrutinise, envision, and question current developments. Whereas I experience the same excitement as Balla painted more than hundred years ago, I hope that I will not be naive. I hope that what I have learnt allows me to better value tomorrow's innovations, improve the discussion of it, and that these innovations will end up in the right place.

[illegible]

J.J. Scholten

Acknowledgement

I am deeply grateful to Dr. Ing. Jens Kober and Dr. Carlos Celemin, who have been my supervisors during the last year. First of all, they inspire me with the quality of their research. And when it came to my thesis work, I have always felt very welcome to ask questions and discuss things, as much and as long as desired. It is this outstanding patience and care that made me feel privileged, and it has moreover been a pleasure to have been challenged and continuously shown strengths and weaknesses in my work.

My friend Daan has been invaluable during my time in Delft. It is because of his everyday support, our numerous discussions about each and every aspect of our studies, about everything else, the thousands of coffees that we had, his brightness, and his humour, that I have always wanted to do hat we have did. Daan, I hope that all your future endeavours will be great, and you will enjoy them as much as I enjoyed your company in Delft.

A special thanks goes out to Rik for his advice on the design and making of graphical content, which has improved the matter far beyond my engineering eye could see.

Last but certainly not least, I sincerely thank my beloved parents and sister for their unwavering and everlasting support.

Contents

1 Introduction

Many imagine a future with flawless autonomous driving where traffic fatalities belong to the past. We expect intelligent drug dosing schemes that minimise adverse effects whilst personal robotics alleviate the chores of our day-to-day life, or we could envision how goods and data are optimally routed such that the planet is relieved from needless pollution and we are saved avoidable waits.

All of the above promises may one day be delivered upon by advances in technology in general and artificial intelligence in particular. A key technique here is Reinforcement Learning (RL), an autonomous learning paradigm where interaction is the driving force. Learning may start from nothing: no models, no assumptions and no pre-programming. Being free from models or even notions such as gravity, velocity or damage makes RL applicable to virtually any sequential decision problem — this alone already makes it such an interesting technique. The algorithm operates by collecting knowledge on the problem, that is perceived somehow (e.g. as sensor data or video), manipulated by any means available, and reflected upon through a (given) performance measure. As such, a RL agent is able to adjust its strategies when faced with modifications in the environment such as a changing nature of the problem or new possible solutions. The RL approach suits control and scheduling problems including those posed in the first paragraph, all of which have partial solutions that are currently approached with RL[1]. Specifically interesting applications are those in which model-based control design is inconvenient or even intractable, such as settings with high-dimensional input/output spaces, or machinery subject to wear and tear (Kober et al., 2013).

In essence, reinforcement learning employs statistical principles to learn and as well as in statistics, quality is driven by quantity. State-of-the art results on application of RL to computer games indeed require up to hundreds of million samples (Horgan et al., 2018). In real-world applications that are constrained in interaction, either physically or time-wise, this requirement renders the problem infeasible. So whilst the tabula rasa property makes RL a strongly generic technique, it meanwhile poses a major weakness in its applicability. When we want to advance reinforcement learning research and the application of it, a primary concern is to ease the restrictive need for interaction (Rastogi et al., 2018).

In this work, we will counter poor sample efficiency by incorporating human insight in the RL learning process, such that the ignorant learning stage is circumvented and the learning agent efficiently attains desired performance. First, let us stipulate some man/machine differences and see why human feedback is especially suitable for the desired acceleration of (deep) RL. In contrast to autonomous learning algorithms, humans are very effective in identifying strategies when faced

[1] Reinforcement learning was applied to robust autonomous driving by Ferdowsi et al. (2018). Treatment with optimal drug schemes for HIV has been studied by Ernst et al. (2007), whereas Zhao et al. (2009) apply these techniques to dose chemotherapy treatment of cancer patients. Robotics and reinforcement learning are discussed by Kober et al. (2013); IFR (2017) and for the context of logistics we refer to Camhi (2018).

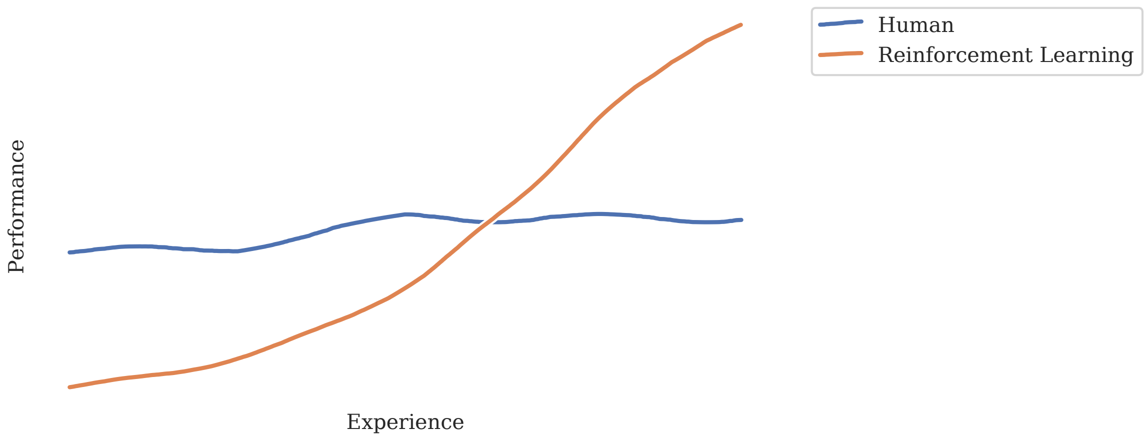

Figure 1.1: A hypothetical illustration of how human insight relates to Reinforcement Learning in terms of potential performance (this may of course vary wildly, depending on the problem). Although computers can be faster and more precise, they need a lot of interaction with the problem before these advantages enter into force. In this study, we aim to combine these properties and have the best of both worlds.

with new problems. As depicted in Figure 1.1, in many cases we achieve decent performance in the first try despite having a poorer precision and reaction time (which may limit us in eventual performance). Indeed, from the sample efficiency perspective, human and RL performance are complementary and incorporating human insight into a learning algorithm is a great way to alleviate one of its major deficiencies.

Throughout this study we focus on continuous control problems and aim for a solution that is scalable and applicable to a wide set of RL baselines in the rapidly advancing field. From these objectives we discuss the literature and then stipulate our understanding of how the combination of RL and human feedback is best approached. These ideas materialise in two algorithms: Probabilistic Merging of Policies (`PMP`) and Predictive Probabilistic Merging of Policies (`PPMP`). Both employ binary corrective feedback as a general and intuitive manner to incorporate human intuition and domain knowledge in model-free machine learning.

The methods estimate uncertainty in the policy, such that corrective feedback can be combined directly in the action space as probabilistic conditional exploration. `PPMP` then extends on `PMP` by predicting the corrected actions. This improves the memorisation of the obtained feedback, such that it has greater effect for lesser corrections. This work is validated both with simulated feedback (for consistent comparison) and actual human participants. For the performance assessment, we use simulated continuous control benchmarks of the OpenAI Gym (Brockman et al., 2016). We will see that the greatest part of the otherwise ignorant learning process can indeed be circumvented. Next to the desired improvements in sample efficiency, our methods enhance final performance, are feedback efficient and robust to erroneous feedback.

In the next chapter, we will discuss the background information and literature upon which this thesis builds. In Chapters 3 and 4 we introduce the two new algorithms. These are validated according to the procedures described in Chapter 5. The results of these experiments are presented and discussed in Chapter 6, and this work than concludes in Chapter 7.

As additional results, we present scalability and sensitivity studies in Appendix A. Individual results from actual human feedback are contained in Appendix B. This work has a accompanying paper (Scholten et al., 2019, see Appendix C) that has been submitted to the 58th IEEE Conference on Decision and Control (CDC).

2 Background and Related Work

This work is rooted in the fields of Reinforcement Learning (RL), Deep Learning (DL), and Interactive Reinforcement Learning (IRL), of which the preliminaries are covered in this chapter. Section 2.1 will set out the principles of RL along with the concept of a Markov Decision Process (MDP) and corresponding assumptions. Second, it is summarised how artificial neural networks can be used as function approximators in Section 2.2. Section 2.3 discusses a combination of the two: the Deep Deterministic Policy Gradient (DDPG) algorithm. Finally, Section 2.4 regards various aspects of human feedback, such as the existing work and developments, how it can accelerate RL and lastly the imperfections that can arise when transmitting knowledge.

2.1. Reinforcement Learning

Reinforcement Learning is a branch of machine learning concerned with experience-driven and autonomous problem solving. As opposed to model-based control approaches, the greater deal of contemporary RL learns by trial and error, without any knowledge or assumptions on the problem at hand (Sutton et al., 1992). The process starts tabula rasa and it is the experience that constitutes the outcome of the learning process. As a corollary, RL is a very general solution that and furthermore has the ability to preserve optimality of the policy in nonstationary situations (for example wear and tear), in contrast to classic control theory (Kober et al., 2013, p. 1250).

Besides naming a class of algorithms, 'RL' also refers to the conventions of the problem these algorithms aim to solve. Central to this framework is the Markov Decision Process (MDP) which defines the quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ (Watkins, 1989, Ch. 3). This set consists of state-space $\mathcal{S}$, action-space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0,1]$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ and constant discount rate γ (Sutton and Barto, 2018, p. 38). RL further defines the following concepts, with reference to Figure 2.1:

Environment If one would explain the RL concepts with a board game as example, the environment would be the board, the playing pieces and the rules of the game. Formally, it constitutes the setting of the problem by defining $\mathcal{S}$, $\mathcal{A}$ and the probability distribution of ending up in a new state s': $\mathcal{T}(s' \mid s, a) = \mathrm{P}\big(s_{k+1} = s' \mid s_k = s, a_k = a\big)$. The environment is subject to time (in this work we use discrete time-index k), and there may also be initial and/or terminal (final) states s_0 and s_f.

Agent Controlled by the reinforcement learning algorithm, the agent interacts with the environment by deciding over possible actions a, given the state s of the environment. This provides for a new state (possibly the same or terminal), and the agent obtains an immediate reward r. The sequence of rewards is used by the agent to learn its optimal policy.

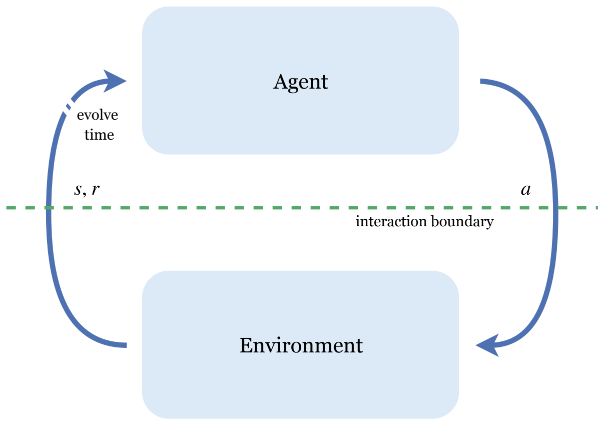

Figure 2.1: The Reinforcement Learning framework defines interaction between agent and environment. Based on the current state, the agent is to decide on an action that is optimal with respect to the reward it obtains.

Markov Property A fundamental assumption in RL is conditional independency of the future with respect to the past, given the current state and input sequence. In other words, all system-dynamic information is contained in the state.

Reward Function The reward function $\mathcal{R}(s, a, s')$ returns a scalar value r, the *instantaneous* reward that reflects how desirable the current situation is with respect to the problem at hand. The ultimate goal of the agent is to maximise the accumulated discounted reward $R_k = \sum_{i=k}^{k_e} \gamma^i r_{i+1}$ with $\gamma \in [0,1]$ being the discount rate and k_e the end of the possibly infinite horizon.

Policy Policy π defines the probability of action a given the current state: $\pi(\mathrm{s}) = \mathrm{P}(a = a_k \mid s = s_k)$. Loosely speaking, the optimal policy $\pi^*(s)$ is the solution to the problem. It is possible to use a deterministic policy instead of a distribution.

Value Function Future rewards are possibly stochastic and R_k thus uncertain. However, given policy π which is followed, it is possible to infer the expected return $\mathbb{E}[R]$ or *value* for every state using the state-value function

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[R_k \mid \pi, s_k = s] \\ &= \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma r_{k+n+1} \,\middle|\, \pi, s_k = s\right]. \end{aligned}$$

Likewise, the expected return of a certain state-action pair may be represented by the action-value function

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}[R \mid \pi, s_k = s, a_k = a] \\ &= \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma r_{k+n+1} \,\middle|\, \pi, s_k = s, a_k = a\right]. \end{aligned}$$

For a great overview of the RL field, (Sutton and Barto, 2018) is recommended, whilst more specific topics are bundled by Wiering and Van Otterlo (2012). In Section 2.3, this writing discusses `DDPG`, a specific algorithm for RL problems. First, let us elaborate about *deep learning*, a technique adopted by `DDPG`.

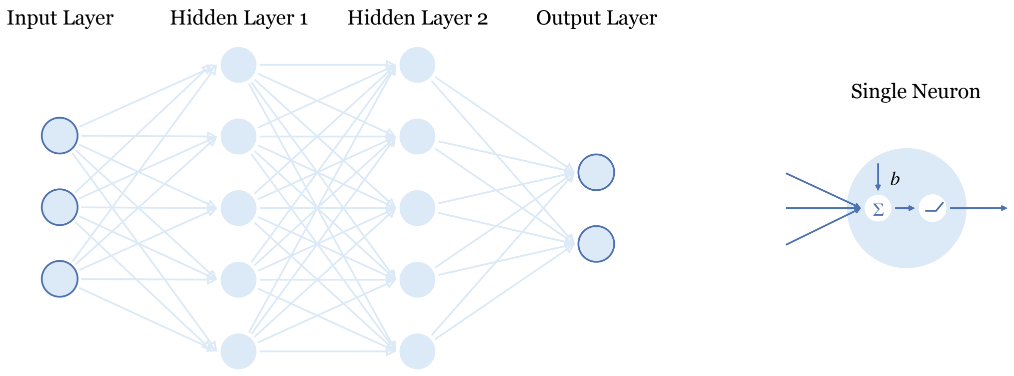

Figure 2.2: A Neural network (left) where arrows are weighed connections. Neurons (right) have a bias b.

2.2. Function Approximation with Neural Networks

Deep Learning (DL) is a field that studies function approximation, inspired by biological neural networks such as sensory neurons in the human eye that connect to the central nervous system and possibly to motor neurons. It is the specific topology of interconnections that is central to the functionality of the network. Different input can activate different regions and lead to a variety of interpretations and reactions. Such functionality is roughly approximated in artificial neural networks, where connections have weight $w \in \mathbb{R}$ and all connections to a neuron are summed with bias $b \in \mathbb{R}$ (w and b are the network's parameters). The resulting signal is saturated by means of an activation function that allows the neural network to adopt great nonlinearities. As in biology, there are many topologies. Most designs organise neurons in layers, where the *layer size* specifies the number of neurons per layer and the *network depth* corresponds to the number of hidden layers. It is customary to coin all learning approaches with multiple hidden layers 'Deep'. This study is concerned with feedforward neural networks, i.e. neurons are only connected to subsequent layers, as in Figure 2.2.

Upon initialisation, the parameters of the network can be optimised to approximate a function using available input-output data. This is called *training*, an iterative process where for each of N input samples the predicted output $\hat{y}_i$ is compared to the true output y_i. Then the parameters are altered to minimise a loss function, such as the mean squared error $J = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$. The de facto standard approach to this optimisation problem is to obtain a gradient by *error backpropagation*, a reverse computation (starting with the output error) respecting the chain rule. The weights in the network are then updated according to $w_{k+1} = w_k - \alpha\frac{\partial J}{\partial w}$ (gradient descent). However, because training is sample-based, hence the gradient stochastic, it can cause for great variance. As a countermeasure, it is commonplace to compute an average gradient over samples in a minibatch (from the greater dataset), such that learning signals that generalise across the minibatch are retained whereas individual errors can be cancelled. On the contrary, low variance may rise a second problem in stochastic gradient descent: getting stuck in local minima. As a tradeoff, minibatches are kept small (typically of size 64) (Lillicrap et al., 2016; Haarnoja et al., 2018).

Besides scheduling the training data, state-of-the-art solvers handle variance and local minima by improving on gradient estimation. For example the family of adaptive moment estimation optimisers (Kingma and Ba, 2014) not only average the gradient over a batch but also over time. In addition, the variance of the gradient estimate is accounted per parameter and inversely proportional to this parameters individual learning rate. In effect, `Adam` downplays updates on parts of the parameter space that have a fluctuating gradient and vice versa. An advantage for convergence is the automatic annealing of the learning rate upon stabilisation of the gradient estimates.

In this study, neural networks are used to estimate knowledge about the problem. The next section elaborates on this application by summarising `DDPG`, a seminal work in the field of deep reinforcement learning for continuous actions.

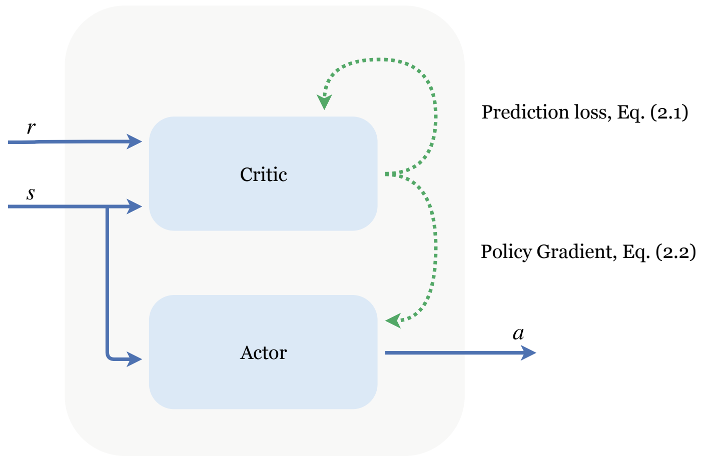

Figure 2.3: Schematic of an actor-critic agent that decides on action a by learning from s and r. The dashed lines indicate how the critic updates both itself and the actor.

2.3. Deep Deterministic Policy Gradient

DDPG, as proposed by Lillicrap et al. (2016), is a popular algorithm that uses neural networks to learn Q and π functions. The actor-critic architecture is derived from the Deterministic Policy Gradient (DPG) method in conventional RL (Silver et al., 2014). In actor-critic learning, independently proposed by both Konda and Tsitsiklis (2000) and Sutton et al. (2000), a value-function and policy are learned simultaneously by the critic and actor respectively (see Figure 2.3). Whilst possibly continuous actions are efficiently obtained (line 5 in Algorithm 1) by a single evaluation of the policy (in contrast, critic-only approaches require $\arg\max$ sampling), the algorithm is still able to learn off-policy, i.e., from transitions that do not necessarily adhere to the current policy (line 8).

Whereas the idea of adopting neural networks in RL is straightforward, the actual implementation is problematic because of instability (Tsitsiklis and Van Roy, 1997). As originally proposed by Mnih et al. (2015), DDPG uses *target networks* and *experience replay* as countermeasures. Originating back to 1992 (Lin), experience replay is a proven effective means of improving sample efficiency. Samples, represented by quadruples $(s_k, a_k, s_{k+1}, r_{k+1})$, are stored and reused to learn recurrently using a *replay buffer* (see line 7 and 8 of Algorithm 1). With respect to stability, experience replay is especially helpful in deep RL because it breaks temporal correlation between samples.

Introduced by Mnih et al. (2015), the target network Q' acts as a copy of the Q-network that is temporally frozen (exempt from updates). It allows to compute the TD-error more consistently as the comparison is not compromised by the possibly high fluctuation of the online Q-estimates. DDPG applies a 'soft update' $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$ to establish tracking with a rate τ (lines 12 and 13). The same technique and notation is used for the policy network with parameters ψ.

The critic network is trained by minimising its prediction error (the mean square of the temporal difference errors, line 10)

$$J^Q = \frac{1}{N}\sum_{i=1}^{N}\left(r_i + \gamma Q'\left(s_{i+1}, \pi'(s_{i+1} \mid \psi') \mid \theta^{Q'}\right) - Q(s_i, a_i \mid \theta^Q)\right)^2$$

where i is an index for the samples in the replay batch. The actor is trained with a gradient from the critic (line 11), such that the policy is altered in the direction of increasing value:

$$\nabla_\psi J^\pi \approx \frac{1}{N}\sum_{i=1}^{N}\left(\nabla_a Q(s, a \mid \theta^Q)\big|_{s=s_i, a=\pi(s_i)} \nabla_\psi \pi(s \mid \psi)\big|_{s=s_i}\right). \tag{2.1}$$

Algorithm 1 Deep Deterministic Policy Gradient (Lillicrap et al., 2016)

```
1: Initialize:
       Neural network parameters $\theta^Q, \theta^{Q'}, \psi, \psi'$
       Replay buffer $B$
2: for episode = 1 to $M$ do
3:     Initialize:
       Ornstein-Uhlenbeck process $v$
4:     for t = 1 to $T$ do
5:         $a_t \leftarrow \pi(s_t|\psi) + v_t$
6:         Obtain $s_{t+1}$ and $r_t$ by executing $a_t$
7:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $B$
8:         Randomly sample $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $B$
9:         Compute target $Q$-values $y_i = r_i + \gamma Q'\left(s_{i+1}, \pi'(s_{i+1} \mid \psi') \mid \theta^{Q'}\right)$
10:        Update $\theta^Q$, minimising loss $J^Q = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q(s_i, a_i \mid \theta^Q)\right)^2$
11:        Update $\psi$ using the sampled policy gradient:
               $$\nabla_\psi J^\pi \approx \frac{1}{N}\sum_{i=1}^{N}\left(\nabla_a Q(s, a \mid \theta^Q)\big|_{s=s_i, a=\pi(s_i)} \nabla_\psi \pi(s \mid \psi)\big|_{s=s_i}\right)$$
12:        Update target network of $Q$:   $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
13:        Update target network of $\pi$:   $\psi' \leftarrow \tau\psi + (1-\tau)\psi'$
14:    end for
15: end for
```

For exploration with temporal consistency, an Ornstein-Uhlenbeck (OU) process (a mean-reverting random walk) v is added to the action (line 5).

2.4. Learning from Human Feedback

Autonomous RL methods can be extended to learn from humans. Such methods, known as Interactive Reinforcement Learning (IRL), bring several benefits. Whilst regular RL is especially powerful in fine-tuning performance (even beyond human abilities) these experience-driven methods lack insight and require many more interactions than humans to learn (Mnih et al., 2015). The required amount of training is especially restrictive in real-world applications (explained in Kober and Peters, 2012) and human feedback can help to overcome this problem (Suay and Chernova, 2011; Amershi et al., 2014; Celemin et al., 2018). Subsequently, RL techniques can account for performance refinements, such that IRL benefits both from human insight, and responsiveness and precision of a computer.

The next subsection categorises three ways to convey information from a teacher to a learning agent. The discussion includes methods that learn from feedback only (without RL-parts), since a great deal of IRL research is rooted there. Next, we will address the leading question of how to combine the information from the feedback with the learning outcome of the agent, such, that the whole is greater than the sum of its parts.

2.4.1. Obtaining Human Feedback

There are many ways to convey knowledge, and some are more informative than others depending on the situation. We will address three methods, sorted by general information richness. First evaluative feedback, then preference-based feedback and finally corrective feedback.

Evaluative Feedback

As evaluative feedback we consider the cases in which the feedback signal is scalar ($h \in \mathbb{R}$) and reflects the quality of what is perceived. For example, one may intuitively provide feedback according to a 5-star rating system (Vollmer and Hemion, 2017). Naturally, these approaches typically interpret such feedback as *reward*, although we will later discuss that this might be suboptimal.

Since the introduction of evaluative IRL by Thomaz and Breazeal (2006), a breakthrough was achieved by Knox and Stone (2009) who designed a general approach called Training an Agent Manually via Evaluative Reinforcement (`TAMER`) — at the time not combined with RL but learning from human feedback only. The main principle in `TAMER` is that human feedback is obtained as a scalar value (similar to MDP reward), and modelled as a function of the state-action space: $H : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. This model is then learned. Innovations of the `TAMER` framework include integration with RL (Knox and Stone, 2012), adoption of the actor-critic architecture (Vien and Ertel, 2012) and extension to Deep Learning (Warnell et al., 2017).

An advantage of evaluative feedback is the intuitive and easy communication. This approach is used in many robotic applications, e.g., León et al. (2011); Knox et al. (2013); MacGlashan et al. (2017); Vollmer and Hemion (2017). However, the ease trades of with the richness of the information, and the benefits scale poorly as problems become more complex (Suay and Chernova, 2011).

Preference-based Reinforcement Learning

In our categorisation we consider *preference based RL* as the case where a teacher selects the best alternative between multiple state trajectories (Wirth, 2017) (it may as well be considered evaluation). Discriminating between suggestions typically results in a better sample efficiency compared to evaluative feedback using scalar signals, because it can be more informative and the information can be related to a greater part of the policy. Indeed, recent work in this direction has proven its effectiveness also for complex environments (Christiano et al., 2017; Pinsler et al., 2018).

From a practical perspective, disadvantages of this method are that it is ideally conducted as side-by-side comparison, which requires simulation and segmentation. Furthermore, ambiguity may arise for demonstrations with distinct suboptimalities and indifferent performance. Still, this feedback is reported to be more robust than evaluative feedback (Pinsler et al., 2018).

Corrective Feedback

The third means of information transfer from teacher to agent discussed in this study is *corrective feedback*, a signal in the action space. A paramount distinction compared to evaluative or preference feedback is that the feedback now concerns what *should* happen (given the situation) rather than what has happened. Corrective feedback is particularly suitable to update a policy in a continuous action space. Conceptually, it allows for pruning of the search tree (space), as opposed to evaluative feedback that rather concerns single nodes (points).

For a discussion on corrective feedback for discrete action spaces, we refer to Chernova and Thomaz (2014, p. 59). Generalisation to a continuous action space was achieved by Celemin and Ruiz-del Solar (2015a,b) who introduced the COrrective Advice Communicated by Humans (`COACH`)-framework. `COACH` collects unitary feedback in the action-domain ($h \in \{-1, 0, 1\}$ for all channels in a) and learns a model that represents and generalises the feedback as a function of the state-space.

In line with the explanation of corrective feedback as a rich source of information, Celemin and Ruiz-del Solar (2018) demonstrate how `COACH` outperforms regular RL, teleoperation and `TAMER` in various continuous settings such as robot dribbling, and simulated bicycle balancing. Suay and Chernova (2011) highlight how corrective feedback especially flourishes in larger problems, an observation experimentally verified in Celemin et al. (2018) also for continuous action spaces.

The freedom in providing corrections may become a difficulty when the action space grows too large and/or the teacher becomes unfamiliar with the action domain. In such cases, mappings between policy space and feedback space can be used (Celemin et al., 2018).

2.4.2. Processing Human Feedback

Although there are different ways in which information from feedback and the learned policy may be combined, it is in general not clear which method is most effective, or how the answer to this question would depend on the different feedback formats. In literature this issue is not broadly covered, with exemption of the two studies reviewed below.

Using evaluative feedback, Knox and Stone (2010) studied eight implementations that use a model $H(s,a)$ of the human reward (as in `TAMER`). Some directly alter the Q-function, or append the signal to the feature vector in order to learn how the feedback should be processed. The best results were found for what is later coined *action biasing* and *control sharing*. In action biasing, $a_k = \arg\max_a Q(s_k,a) + B(s_k,a)H(s_k,a)$ with B a tunable, decayed state-dependent scaling function. In control sharing, B is used to alternate between policies according to a stochastic approach where $\mathrm{P}\big(a = \arg\max_a H(s,a)\big) = \min(B(s,a),1)$. Note that the two sources of knowledge are iterated between and do not supplement each other: suggestions are followed regardless possible inconsistency with what is already learned, and vice versa. Interestingly, both methods directly affect action selection, and Q itself is not modified.

Building on these observations, Griffith et al. (2013) emphasise how human feedback is given with respect to the policy, and indeed should affect action selection directly rather than learning of value estimates. They propose to use distributional RL and derive the Bayes-optimal multiplication of policies: $\pi \propto \pi_{\mathrm{RL}} \times \pi_{\mathrm{T}}$, where the latter two policy notations represent conventional RL and the teacher respectively. The probabilistic approach, they reason, is effective in handling uncertainties of the received feedback. Using a synthetic oracle to simulate feedback and a consistency estimate to cope with erroneous feedback, significant improvements in learning progress were reported (comparing against action biasing and control sharing). These results have been recognised by Cederborg et al. (2015), who verified them with actual participants.

A combination of RL and *corrective* feedback is proposed in Celemin et al. (2017, 2018), who map corrective feedback to the feature space of an actor-only RL baseline, and use it as additive noise to obtain an alternative policy that is consistent throughout the episode. Using policy-search (actor-only) baselines, their method is able to learn motor primitives for swingup, drawing and bilboquet problems.

2.4.3. Difficulties of Human Feedback

Whilst learning with human feedback can accelerate RL, it will also introduce certain flaws that stem from human nature. An important problem can be feedback inconsistency (Griffith et al., 2013) as a result of mistakes or a changing feedback policy. Furthermore, certain types of feedback (or ways to provide it) may suit the teacher and/or problem better than others (Thomaz and Breazeal, 2008) such that the feedback quality may be further compromised in case of a poor fit.

Accounting for inconsistent human feedback is done in various ways across literature. Peng et al. (2015, 2016) process feedback with a bias for teacher personality (e.g. reward or punishment-based). Griffith et al. (2013) increase robustness by means of a consistency estimate that adaptively schedules the influence of the feedback, such that inconsistent information can be downplayed. This strategy is adopted by Lin et al. (2017) in the context of a deep RL study.

Other distortions may arise in the form of extreme feedback values, presumably a result of human simplification when operating a slider (Senft et al., 2017). Furthermore, Vollmer and Hemion (2017) discuss how people may have different feedback strategies or social inclinations, illustrated with examples of people that intend to motivate the agent by giving extra reward. This optimistic feedback resulted in premature convergence, whereas Thomaz and Breazeal (2006) studied a 'guidance' approach that actually relies on such future-directed feedback. Last but not least, teachers will be quiet allegedly 30% of the time (Cederborg et al., 2015), and this silence may mean different things, and not necessarily optimality; the teacher could be inconclusive, neutral or simply distracted.

2.5. Conclusion

In this chapter, the concept of the Markov Decision Process was established along with the Reinforcement Learning framework. In succession it was discussed how Deep Learning allows for general function approximation and generalisation, and how this technique applies to RL by regarding the `DDPG` algorithm. Next, it was discussed how (human) experience and insight may be carried over to a learner online either by evaluative, comparative or corrective feedback (in increasing order of information-richness and possibly combined). Whilst the first options are better researched, the latter is especially promising when applied to larger problems and continuous action spaces, because corrective feedback contains most information in most situations. In addition, we have presented evidence that it is most effective to apply human feedback in action selection. Last but not least, it was argued that a distributional approach can be effective to cope with the uncertainties that arise from the feedback, and moreover the errors and inconsistencies stipulated in the penultimate section of this chapter.

Yet, a method that combines *deep* RL with corrective feedback does not exist, and that brings us to the coming chapters.

3 Proposed Method 1: Probabilistic Merging of Policies

We here establish the principles behind our work with the intention to answer the key question of how human intuition is best used to improve the sample efficiency of deep Reinforcement Learning (RL) for continuous control problems. Central to this objective is our observation that humans are very effective in identifying strategies and sometimes achieve decent performance even in the very first try, but with limited precision, whereas computational agents have opposite properties — complementary, in the light of efficient learning.

In Section 3.1, we stipulate the motivation of our approach with reference to existing literature. In succession, Section 3.2 covers the realisation of these ideas by presenting the *selector* module, the multihead actor network and a schematic overview. This chapter is concluded in Section 3.3.

3.1. Motivation and Related Work

Our method is founded on a set of insights as derived both from literature and our own view on the matter. Partly, these propositions have been covered in the previous chapter. As illustrated in Figure 3.1, we enumerate the following ideas for future reference

1. **Action Selection** After first evidence by Knox and Stone (2010, 2012), it later was Griffith et al. (2013) who made a strong case for how feedback is given with respect to the action (sequence) and it is most effective to directly adjust the actions in case feedback is obtained. In a comparison against other evaluative feedback methods, their algorithm outperformed all baselines that instead affect the actions indirectly by modification of the policy (Section 2.4.2).

2. **Significant Error** If we consider the early learning phase, where the policy is useless but the feedback most valuable, we believe feedback is received in case of a significant error as to help the agent develop a notion of the task rather than to communicate refinements. Moreover, we argue that the instantaneous precision of feedback is rather coarse (instantaneous, as in contrast to feedback on steady-state error in a later learning phase). We quantify this limitation by defining a precision d that expresses a region of indifference per dimension. d is used as a radius, so in a 1D-example, let $d = 1$ and feedback h on policy $a_p = -1$ thus only be given in case the optimal action $a^* < -2$ or $a^* > 0$.

3. **Reinforcement Learning for Fine-stage Optimisation** Reinforcement learning is superior in final performance because of its precision and reaction time (Mnih et al., 2015). From our

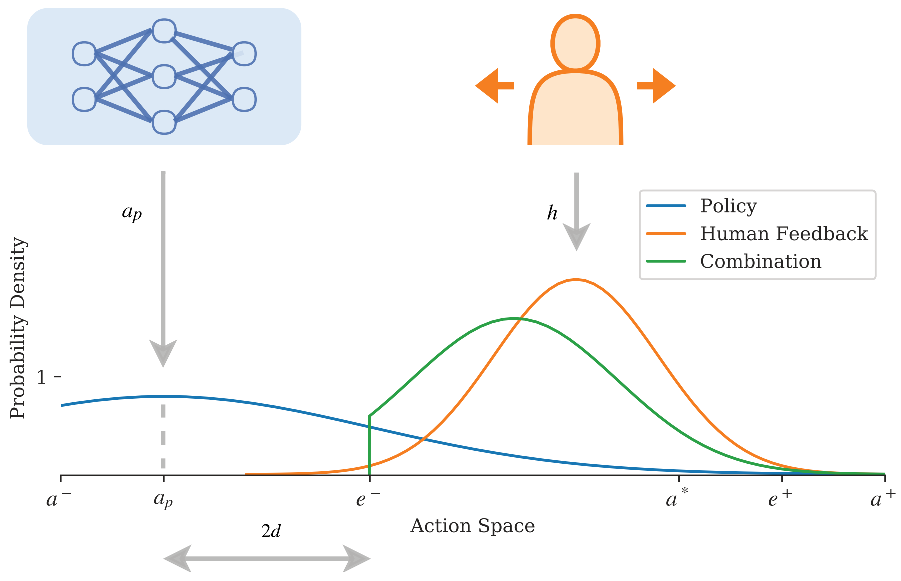

Figure 3.1: Using respective covariances, the policy is combined with human feedback in the action space. The resulting distribution on the action that is selected, is truncated such that corrections always have significant effect and the given information cannot dissipate in case of an overconfident policy.

point of view, it should therefore be allowed to autonomously optimise during the later learning phases, such that local optima are identified (e.g. using gradients) independent from past feedback.

4. **Probabilistic Approach** In the words of Losey and O'Malley (2018): *"When learning from corrections ... [the agent] should also know what it does not know, and integrate this uncertainty as it makes decisions"*. We subscribe this point of view, and furthermore emphasise past success of this principle in other fields (Thrun et al., 2005). However, whereas Losey and O'Malley (2018) then estimate the uncertainty in the corrections, we consider feedback covariance fixed (related to the precision d in Item 2, **Significant Error**) and argue that the correction size is inversely proportional to the learning progress (performance) of the agent. We combine this view with the work of Griffith et al. (2013), who proposed to combine the policy and feedback distributions to determine the action.

5. **Corrective Feedback** In line with the discussion in Section 2.4.2, we choose to comply with binary corrective feedback as a scalable, feasible and powerful input format that is especially suitable for continuous action spaces (Celemin and Ruiz-del Solar, 2015b). Nevertheless, our approach is ready for use with corrective feedback of any real number, so this proposition is not restrictive, but merely a framework.

It immediately becomes apparent that some of these ideas align, e.g. that corrections are inaccurate (2), but do not need to be accurate, since RL will efficiently identify local minima (3). However, before connecting the dots, let us complete this motivation by stipulating the following assumptions:

1. Given the assumed area of indifference of Item 2 (**Significant Error**) expressed by d (Figure 3.1) the RL functionalities are able identify the local minimum within this area. In other words, the Items 2 and 3 concern overlapping regions. As an implication, the global optimum is attained if the feedback brings us anywhere close.

2. Over time, the magnitude of the error e between the policy and the optimal action will show a diminishing trend.

3. We are able to estimate the covariance of the policy online and this covariance will diminish as the agent learns.

As a corollary of the above statements, we develop a learning method where actions are obtained by significant modification of the policy with respect to obtained binary feedback, in a probabilistic manner that reflects on the current abilities of the agent. A natural advantage of this approach is that it preserves autonomy and optimality (Item 3, **RL for Fine-stage Optimisation**), as there will be no feedback when the teacher deems the performance satisfactory (Assumption 2) and our approach then recedes to its RL principles.

There is further evidence in literature that supports our approach at least partly but has not yet been mentioned. In an autonomous (regular) RL study, Haarnoja et al. (2018) successfully combine the probabilistic argument with vigorous (maximum entropy) exploration. The idea of using feedback as exploration is recognised by Nair et al. (2017) however their method requires full demonstrations and post-processing these in simulation. Last but not least, Ngo et al. (2014) demonstrate uncertainty-handling exploration for classification problems aided by binary human feedback. They show an improvement in sample efficiency, robustness to errors and that for negligible extra computational costs.

3.2. Design

This section describes how the aforementioned ideas materialise in the learning algorithm that we coin Probabilistic Merging of Policies (PMP). First, it is established how the feedback and policy information is reconciled by comparing respective covariances. The second section covers how the modified action selection process is incorporated in the complete learning framework. Last, we discuss how to infer the policy covariance by means of a multihead neural network.

3.2.1. Combining Policy Information in the Selector

For the sake of this explanation, let us temporally assume a one-dimensional situation with non-erroneous directional feedback $h \in \{-1, 0, 1\}$ on current policy a_p that indicates the relative location of optimal action a^* (observe Figure 3.1). In line with Item 1 (**Action Selection**), our approach is to immediately alter the policy a_p in the direction of h, such that the eventually selected action

$$a = a_p + \hat{e}h \tag{3.1}$$

(the orange distribution in Figure 3.1) and $\hat{e}$ being an estimate of the absolute error $|a_p - a^*|$.

Inspired by the principles of the Kalman filter, we estimate the unknown magnitude of the error using the covariance of the policy (the prediction) and the feedback (an observation) in a module that we call the *Selector*. It is assumed that the greatest errors occur during the early stages of the learning process, and these will diminish over time along with the covariance of the policy $\Sigma_{a_p a_p}$. With the covariance of the feedback Σ_{hh} as a known constant (Item 2, **Significant Error**), and $\Sigma_{a_p a_p}$ obtained as described later in Section 3.2.3, let

$$\begin{aligned} \hat{e} &= G\,\mathrm{diag}(c_s) + \mathbf{1}c_o^T \\ \text{with } G &= \Sigma_{a_p a_p}(\Sigma_{a_p a_p} + \Sigma_{hh})^{-1} \\ \text{and } \mathbf{1} &= [1, 1, \ldots, 1]^T. \end{aligned} \tag{3.2}$$

(see lines 8 and 9 in Algorithm 2) where the constant vectors c set the bounds on $\hat{e}$ as described in the last two paragraphs of this section.

Note that $G \in (0, 1)$ (all Σ are positive definite by definition) is analogue to the Kalman gain as a dimensionless trade-off measure. When the policy shows great covariance the corrections will have

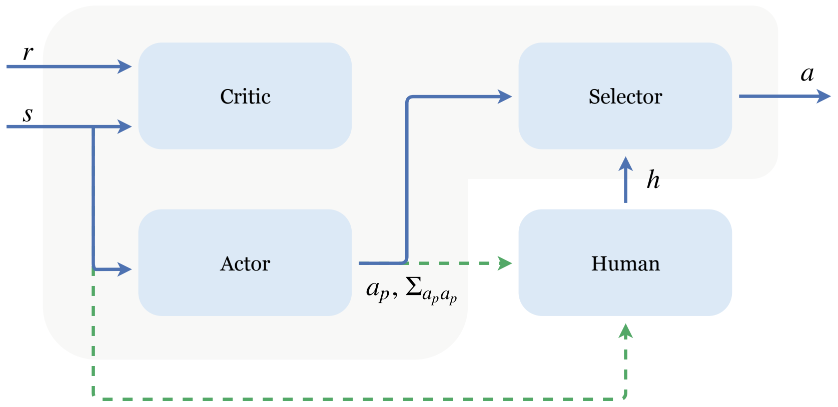

Figure 3.2: Schematic representation of the suggested approach, where human feedback is combined with the policy. The elements in the grey area constitute an autonomous learner.

greater effect and facilitate vigorous exploration, and the inverse: upon convergence, corrections will be more subtle. Besides that the exploration is automatically annealed over time (Assumption 3), its effect is furthermore state-dependent and tailored for every action channel, respecting correlations.

The relation of G to $\hat{e}$ is defined using two vectors, of which the length equals the dimensionality of the action space. First, let us discuss offset c_o. With reference to Item 2 (**Significant Error**), a lower bound on e in terms of the feedback and its precision d is expressed as $e^- = a_p + hd$. Moreover, for the action selection in Equation (3.1) we may further restrict the search by using the fact that a is guaranteed to be closer to a^* than a_p even if $\hat{e} = 2d$. By Assumption 1, it is therefore more efficient to use this as a lower bound on $\hat{e}$, that is farther from a_p. Setting $c_o = 2d$ in Equation (3.2) realises exactly this (see Figure 3.1). Apart from the optimisation perspective, it is always desired to apply a significant correction in case feedback is provided. First, it will avoid frustration of the user. Second, the information is not preserved otherwise.

The scale c_s allows us to set an upper bound e^+ for the applied corrections. From the perspective of using human feedback as exploration, let us consider the case where the policy suggests some negative action, and receives $h = 1$ since optimality is contained in the positive half of the action space (Figure 3.1). Although we cannot make any general statements about the reachability of the state-space, it is clear that feedback can only have the intended effect when $\hat{e}$ is large, else there is no escape from the wrong half of the action space.

3.2.2. Integrating the Selector with Autonomous Learning

The ideas established in the previous paragraph raise requirements for the eventual algorithm. The probabilistic combination of the policy and the feedback results in off-policy data in a continuous action space. As critic-only methods are suitable for a discrete action space whilst actor-only methods are on-policy, an actor-critic scheme remains as the evident choice.

Figure 3.2 illustrates how the policy of an actor-critic scheme is combined with human feedback in the selector to obtain action a. It is assumed that the human provides binary feedback signals h occasionally and bases this on the observed state sequence (and possibly the actions). Delays between perception and feedback are not taken into account. The selector module adjusts a_p with respect to h as described in the previous paragraph. In case feedback is not provided ($h = 0$) the algorithm relies on its own policy, including exploration noise. Autonomy is hereby preserved.

Next, let us see how an estimate of $\Sigma_{a_p a_p}$ is obtained by means of a multihead neural network.

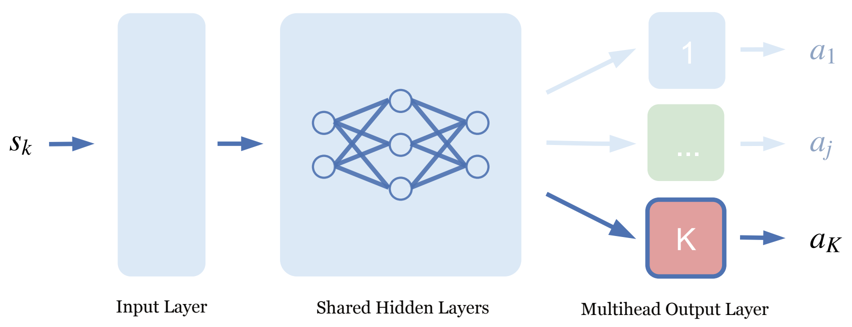

Figure 3.3: A multihead implementation of a neural network with K heads. We use index j to denote a particular head.

3.2.3. Multihead Actor Network

The purpose of the multihead neural network is not only to estimate a quantity but furthermore to estimate the covariance in this estimate. Gal (2016) has established that the uncertainty over a deep neural networks output may be obtained from multiple passes with dropout[1]. Osband et al. (2016) however report how, at least in the context of RL, a multihead neural network that maintains multiple hypotheses is a more consistent approach to generate posterior samples than dropout. Whereas in their study the purpose of the multihead network is to directly use the posterior for exploration rather than to quantify confidence (as desired for our approach), Rupprecht et al. (2017) indeed establish how multiple hypotheses provide for accurate estimation of abilities in a deep learning classification problem. As it is furthermore desired to have efficient and scalable estimation[2], we apply the multihead architecture as discussed in Osband et al. (2016) to the actor network, according to the schematic in Figure 3.3.

Effectively, the modification of a regular actor network to its multihead counterpart results in K copies of the output layer that estimate the optimal action $a_j = \pi_j(s \mid \psi)$, where j indicates the head. For our Gaussian inference, we then estimate $\Sigma_{a_p a_p}$ as $\operatorname{cov}\big(\pi(s_k \mid \psi)\big)$ (line 6 in Algorithm 2). For the training, we establish an extension to Equation (2.1) that features individual values of $\nabla_a Q$ and $\nabla_\psi \pi$ for each head. This sampled multihead policy gradient

$$\nabla_\psi J^\pi \approx \frac{1}{N}\sum_{i=1}^{N}\Big(\Phi\, \nabla_\psi \pi(s \mid \psi)\big|_{s=s_i}\Big)$$

$$\text{with}\quad \Phi = \left.\begin{bmatrix} \nabla_a Q(s,a \mid \theta^Q)^T\big|_{a=\pi_1(s_i)} \\ \nabla_a Q(s,a \mid \theta^Q)^T\big|_{a=\pi_2(s_i)} \\ \vdots \\ \nabla_a Q(s,a \mid \theta^Q)^T\big|_{a=\pi_j(s_i)} \end{bmatrix}^T\right|_{s=s_i} \quad \text{and}\quad \pi(s \mid \psi) = \begin{bmatrix} \pi_1(s \mid \psi) \\ \pi_2(s \mid \psi) \\ \vdots \\ \pi_j(s \mid \psi) \end{bmatrix}. \tag{3.3}$$

The question remains how to determine the policy, needed both during action selection and calculation of the temporal difference error. Note that the policies are possibly multimodal and in such case averaging the individual policies is an unfortunate choice. For a_p (line 5 in Algorithm 2) we choose to randomly select a head j_e per episode such that temporal consistency is preserved. For the training of the critic (line 13), the target policy $\pi'_{j_i}(s_{i+1} \mid \psi')$ is evaluated for j_i, the same head as was used to predict a_i.

Apart from providing for a uncertainty estimate, the functioning of the multihead implementation is no different than a regular actor network.

[1]Dropout, a regularisation technique, refers to the practice of randomly setting weights to zero with a small probability (Srivastava et al., 2014).

[2]In Section A.2 of the appendices it is studies how the computational complexity scales with the amount of heads K.

Algorithm 2 K-Head Probabilistic Merging of Policies (PMP)

```
1: Initialize:
       Neural network parameters $\theta, \theta', \psi, \psi'$
       Replay buffer $B$
       Feedback covariance $\Sigma_{hh}$
       Scale $c_s$ and offset $c_o$
2: for episode $e = 1$ to $M$ do
3:     Initialize:
       Ornstein-Uhlenbeck process $\nu$
       Randomly set active head $j_e$
4:     for timestep $k = 1$ to $T$ do
5:         $a_p \leftarrow \pi_{j_e}(s_k \mid \psi) + \nu_k$
6:         $\Sigma_{a_p a_p} \leftarrow \text{cov}\big(\pi(s_k \mid \psi)\big)$
7:         Obtain feedback $h_k$
8:         $G \leftarrow \Sigma_{a_p a_p}(\Sigma_{a_p a_p} + \Sigma_{hh})^{-1}$
9:         $a_k \leftarrow a_p + \big(G\,\text{diag}(c_s) + \mathbf{1} c_o^T\big)\, h$
10:        Obtain $s_{k+1}$ and $r_k$ by executing $a_k$
11:        Store transition $(s_k, a_k, r_k, s_{k+1}, j_e)$ in $B$
12:        Randomly sample $N$ transitions $(s_i, a_i, r_i, s_{i+1}, j_i)$ from $B$
13:        Compute target $Q$-values $y_i = r_i + \gamma Q'\left(s_{i+1}, \pi'_{j_i}(s_{i+1} \mid \psi') \mid \theta'\right)$
14:        Update $\theta$, minimising loss $J^Q = \frac{1}{N}\sum_{i=1}^{N}\big(y_i - Q(s_i, a_i \mid \theta)\big)^2$
15:        $\Phi \leftarrow \nabla_a Q(s, a \mid \theta)|_{s=s_i, a=\pi_n(s_i)}\ \forall n \in \{1, 2, \ldots, K\}$, see Equation (3.3)
16:        Update $\psi$ using the sampled multihead policy gradient:

               $$\nabla_\psi J^\pi \approx \frac{1}{N}\sum_{i=1}^{N}\left(\Phi\, \nabla_\psi \pi(s \mid \psi)\big|_{s=s_i}\right)$$

17:        Update target network of $Q$:   $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
18:        Update target network of $\pi$:   $\psi' \leftarrow \tau\psi + (1-\tau)\psi'$
19:    end for
20: end for
```

3.3. Conclusion

In this chapter the foundations of our approach have been established and they were connected to existing work. We have established five founding ideas that concern 1) application of feedback in the action domain 2) the magnitude of corrections during the early learning phase 3) the strength of RL for final performance and autonomy 4) the advantage of using uncertainty when reconciling policy information and 5) binary corrective feedback as an intuitive, effective and scalable format. The realisation of this idea was motivated with sections on probabilistic action selection in the *selector* module, obtaining policy covariance by means of a multihead neural network and a section on the complete interconnection. In conclusion, the content of this chapter constitutes a promising and very general idea to accelerate RL problems by means of feedback, along with an implementation that is straightforward and computationally efficient.

What was covered in this chapter will be extended in the next, where a module is introduced that explicitly stores corrected state-action pairs.

4 Proposed Method 2: Predicted Probabilistic Merging of Policies

The previous chapter introduced a general method called Probabilistic Merging of Policies (PMP) that improves the sample efficiency of deep Reinforcement Learning (RL) in continuous control problems. The method uses human corrective feedback to affect action selection and than stores past interaction in a buffer to learn from it. Ironically, the originally poor sample efficiency of DDPG may in this approach also impede the proposed solution to it. When feedback is only contained in a small subset of the interaction stored in the buffer (this is true when the feedback rate is low) its effect is limited, as in deep learning single samples deliberately have little effect. The fundamental instability problem (Tsitsiklis and Van Roy, 1997) will otherwise cause divergence.

In this chapter, we propose a method to generalise the obtained feedback and reuse it during learning. This second algorithm, coined Predictive Probabilistic Merging of Policies (PPMP), engages a copy of the actor network (assuming its architecture and hyperparametrisation is the best known choice) that predicts the corrected actions. Under the assumption that the feedback source's underlying policy is not subject to change, a rough estimate of corrections is now available for every state, even when feedback is not provided. Following the argumentation of the previous chapter, this rough estimate is exactly what we need to improve sample efficiency.

First, the design is explained Section 4.1. In Section 4.2 the consequences of applying what is predicted whilst predicting what is applied are discussed. Section 4.3 concerns the difference in quality between the policy and the corrections, and how the respective performances crossfade over time. Section 4.4 covers the solution to the problems explained in its preceding two sections. The conclusion of this chapter is found in Section 4.5.

4.1. Design

The proposed architecture featuring the *predictor* $P(s \mid \phi)$ is illustrated in Figure 4.1, and the pseudocode is contained in Algorithm 3 with the elements from PMP printed in grey. The predictor has parameter set ϕ and it is trained with corrected samples from a separate buffer B_c (line 25). Now, there are always two actions available before possible corrections are applied:

1. the policy a_p from the actor (line 8)
2. the predicted corrected action $\hat{a}_c$ from the predictor (line 9).

The critic decides which actions is to be executed, and this action is labelled as a_Q (line 10). Before providing the details of this substitution, that allows us to reap the harvest of predicted corrections, two problems that this new module causes need to be addressed.

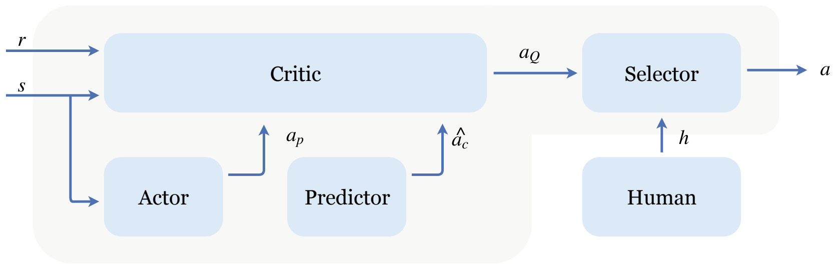

Figure 4.1: A schematic representation of Predictive Probabilistic Merging of Policies (PPMP), where the *predictor* module lessens the required amount of feedback by predicting corrected samples $\hat{a}_c$. The critic then chooses either a_p or $\hat{a}_c$. Human feedback may then be given with respect to this choice a_Q and is is again the selector that then computes the eventually executed action a.

4.2. Failure Caused by Prediction

A naive implementation of the PPMP scheme as presented in the previous section would be to take action $\hat{a}_c$ ‘greedily’ at every step in the early learning stage. As a corollary, the data in the experience buffer B_c would then solely consist of samples that adhere to the predictor policy $P(s \mid \phi)$ (disregarding prediction error). Note that this policy is a generalisation of little data, namely the the corrected samples only.

As clearly established in the work of de Bruin et al. (2015), training on uniform data from a stationary policy destabilises the learning in an off-policy framework. Moreover, this shortcoming seems to specifically concern policies with consistent performance, as these cause for small variation in the replay buffer. The authors reason about overfitting because of the strongly correlated data. In addition, we suspect that the Adam optimiser (Kingma and Ba, 2014) may become overconfident in its gradient estimate (the variance in the sampled gradients drops when the estimates are consistent) and raises at least some of its adaptive learning rates to an unstable value.

The aforementioned problem has been very apparent in the development of PPMP. Although it is possible to have great performance from the start by maximal utilisation of $\hat{a}_c$, this approach is very prone to divergence, although it typically recovers soon once some data of the unstable policy is collected. As we deem it more desirable to have an approximately monotonically increasing performance curve, it is needed to address the decision making between a_p or $\hat{a}_c$.

We introduce three countermeasures that address the variance issue. From our experiments we conclude that it is essential to collect data with a random or pristine policy (e.g. solely the OU-process, or the actor right after random initialisation). Therefore, as a first but partial solution, we enable the predictions in line 9 only after a first training split of N_P samples is collected. It is however still required to alternate between predictions a_p and $\hat{a}_c$ after this ‘cold-start’. We will address this decision in Section 4.4, after discussing a second countermeasure here, and a second problem with the use of $\hat{a}_c$ in the next section.

The aforementioned problem is furthermore countered and partly overcome by injecting noise in the predicted corrected sample, thus $a = \hat{a}_c + \mathcal{N}(0, \sigma_{\hat{a}_c})$, with $\mathcal{N}$ being a Gaussian random vector centred around zero. This noise may be related to the resolution d and the fact that $\hat{a}_c$ has never been more than a very noisy estimate (Item 2, Section 3.1), such that additive noise rather restores the original distribution than disturbing it. Our primary concern however is to alleviate the variance problem in the experience buffer, and for this it is sufficient to have a constant value as a tuning parameter. We set $\sigma_{\hat{a}_c}$ per channel, defined as a fraction of the action range. Alternatively, this can be thought of as local system identification using Gaussian noise, as commonplace in system identification literature (Verhaegen and Verdult, 2007).

4.3. Transcending Performance Limitations

Another adverse effect of taking $\hat{a}_c$ is that over time, the policy will perform favourably as the quality of the estimated corrections is limited. In addition, the quality of the feedback source may fall short of what is attainable with RL. While in the previous section it was argued that the predictor should be disabled in some of the early learning steps, we here conclude that the influence of the feedback again needs to be scheduled away during the later learning phases.

Griffith et al. (2013) anneal the influence of the feedback model by means of a hyperparameter. Although their and our method both account the covariance of the policy, the authors reason that this estimate can converge too quickly and the scheduling needs to be accomplished otherwise. However, from our point of view and the experience of designing an annealing schedule, the choice of annealing heuristic greatly affects the learning curve. When the predictor is disabled too early, the actor may cause for a setback in performance. In contrast, retaining the noisy corrected estimates for too long will suppress performance even though a more optimal policy has become available. The exact intersection is eventually completely dependant on the environment and the feedback given (both quantitatively and qualitatively). Hence, manually tuning for the optimal tipping point is not a great solution, and possibly even infeasible in case of high-dimensional problems.

Another approach is to train the predictor on a mixture of data, as Vecerík et al. (2017); Nair et al. (2017) do with the actor. It would allow the prediction module to converge to the policy over time. However, there is a different discrepancy in this approach, as it is as well the task of the predictor to suggest something that is significantly different from the policy. Especially during the first stages of learning mixing in policy data is therefore of great adverse affect to the prediction quality. Since this is the time where the predictor is needed, we do not subscribe to this approach — not to mention the sampling heuristics that become quite involved (Vecerík et al., 2017).

Our solution to the sampling problem uses value estimates as a general solution that does not require domain knowledge. It is treated in the next section.

4.4. Scheduling the Predictions using the Q-Filter

In the previous two sections it was discussed why it is needed to rely only partially on the estimates $\hat{a}_c$ provided by the predictor, namely to prevent overfitting and to retain autonomy and optimality upon convergence. We accomplish the required scheduling by applying a Q-filter (Nair et al., 2017) that selects actions based on their expected value, and so kill two birds with one stone. Setting $a_k = \arg\max_a Q(s,a)|_{s=s_k, a=\pi_n(s_k) \vee a=\hat{a}_c}$ is the most fruitful heuristic to interleave a with $\hat{a}_c$ and moreover automatically fades the control towards the policy upon its convergence (line 10). We have experienced that the performance using the Q-filter is greater than the sum of its parts, presumably because the relative optimality of either policy, at a given time, depends on the state in which it is evaluated, such that ϵ-style selection violates temporal correlation of preference.

There is however one hitch in the application of a Q-filter, being that the critic needs to be learned first. As a corollary, the first stage of the learning does not benefit from the predictions to the fullest extent, as these predictions are at that stage often unjustly rejected. Although the resulting suboptimality may be acceptable, in this study we particularly aim to improve performance during the earliest learning stage in order to demonstrate the potential of corrective feedback. For that matter, we may also exploit the fact that the critic converges faster than the policy (Konda and Tsitsiklis, 2000; Grondman et al., 2012) and it is advantageous to enable the Q-filter only after a certain amount of training (and until then rely on the predictor). This amount is defined as a hyperparameter $N_Q > N_P$, and it is emphasised that the difference in convergence speed between the critic and the actor offers a wide range in which this number is successfully set — in contrast to the aforementioned heuristics. We provide a hyperparameter study in Appendix A.1.

Algorithm 3 K-Head Predictive Probabilistic Merging of Policies (PPMP)

```
 1: Initialize:
        Neural network parameters $\theta, \theta', \psi, \psi', \phi$
 2: Replay buffers $B$ and $B_c$
 3: Feedback covariance $\Sigma_{hh}$
 4: Scale $c_s$ and offset $c_o$
 5: for episode $e = 1$ to $M$ do
 6:     Initialize:
        Ornstein-Uhlenbeck process $\nu$
        Randomly set active head $j_e$
 7:     for timestep k = 1 to $T$ do
 8:         $a_p \leftarrow \pi_j(s_k \mid \psi) + \nu_k$
 9:         $\hat{a}_c \leftarrow P(s \mid \phi) + \mathcal{N}(0, \sigma_a)$
10:         $a_Q \leftarrow \arg\max_a Q(s, a)|_{s=s_k, a=a_p \vee a=\hat{a}_c}$
11:         Obtain feedback $h_k$
12:         if $h_k$ then
13:             $\Sigma_{a_p a_p} \leftarrow \text{cov}\big(\pi(s_k \mid \psi)\big)$
14:             $G \leftarrow \Sigma_{a_p a_p}(\Sigma_{a_p a_p} + \Sigma_{hh})^{-1}$
15:             $a_k \leftarrow a_Q + \left(G\,\text{diag}(c_s) + \mathbf{1}c_o^T\right) h$
16:             Store $(s_k, a_k)$ in $B_c$
17:         end if
18:         Obtain $s_{k+1}$ and $r_k$ by executing $a_k$
19:         Store transition $(s_k, a_k, r_k, s_{k+1}, j_e)$ in $B$
20:         Randomly sample $N$ transitions $(s_i, a_i, r_i, s_{i+1}, j_i)$ from $B$
21:         Compute target $Q$-values $y_i = r_i + \gamma Q'\left(s_{i+1}, \pi'_{j_i}(s_{i+1} \mid \psi') \mid \theta'\right)$
22:         Update $\theta$, minimising loss $J^Q = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q(s_i, a_i \mid \theta)\right)^2$
23:         $\Phi \leftarrow \nabla_a Q(s, a \mid \theta)|_{s=s_i, a=\pi_n(s_i)}\ \forall n \in \{1, 2, \ldots, K\}$, see Equation (3.2)
24:         Update $\psi$ using the sampled multihead policy gradient:
                $$\nabla_\psi J^\pi \approx \frac{1}{N}\sum_{i=1}^{N}\left(\Phi\, \nabla_\psi \pi(s \mid \psi)\Big|_{s=s_i}\right)$$
25:         Randomly sample $N$ transitions $(s_i, a_i)$ from $B_c$
26:         Update $\phi$, minimising the loss $J^P = \frac{1}{N}\sum_{i=1}^{N}\left(P(s_i \mid \phi) - a_i\right)^2$
27:         Update target network of $Q$:   $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
28:         Update target network of $\pi$:   $\psi' \leftarrow \tau\psi + (1-\tau)\psi'$
29:     end for
30: end for
```

4.5. Conclusion

This chapter introduced Predictive Probabilistic Merging of Policies (PPMP), a method that extends the the PMP algorithm by predicting the corrected samples in order to relieve the need for feedback and ease the user's efforts. Along with the design, we reflected upon two issues raised by the predicted corrections, namely 1) a lack of variance in the experience buffer and 2) the falling behind of performance when not scheduling back to the policy upon its convergence. Subsequently, we proposed to delay the predictor module, inject little noise and schedule between predictions and the policy by regarding the Q-value of the estimates. With these three simple countermeasures, the two problems are overcome. After an explanation of our experiments in the next chapter, the performance of the two algorithms is presented in Chapter 6.

5 Experimental Setting

The proposed algorithms PMP and PPMP will be compared to two benchmarks in the field of deep learning. The first reference implementation is DDPG, the deep RL method that this study builds on (Lillicrap et al., 2016). DDPG is a regular RL algorithm that learns from MDP reward only. The second baseline is Deep COrrective Advice Communicated by Humans (DCOACH) (Pérez Dattari et al., 2018). DCOACH learns from binary corrective feedback, regardless of MDP reward. This chapter covers how this comparison is realised.

First, in Section 5.1, the technicalities of our implementation are described. Subsequently three continuous control problems from the OpenAI Gym (Brockman et al., 2016) are introduced that will be used to validate our approach. In Section 5.3 it is discussed how synthesised feedback is constructed as a consistent means to study the performance of the methods subject to feedback. Thereafter, the experiments with human participants are motivated in Section 5.4. This chapter is brought to a close in Section 5.5.

5.1. Details of Implementation

Our source code is publicly available at github.com/janscholten/ppmp. The training of Q and (multihead) π networks follows the work of Lillicrap et al. (2016), except for the multihead extension presented in Section 3.2.3. For the critic, a neural network is used with two hidden layers of size 400 and 300, where the actions are included at the second layer. The activation functions of the critic network are rectified linear units (Glorot et al., 2011). The actor is implemented using the multihead approach described in the previous section, again with two hidden layers of size 400 and 300 respectively. There are ten output heads that have hyperbolic tangent activation, such that the actions are delimited within their bounds. The other activation functions in the actor network are again rectified linear functions. For all networks except the predictor (Section 4.1) we have target networks with soft updates as discussed in Section 2.3, and we train with the Adam optimiser of Kingma and Ba (2014).

5.1.1. Probabilistic Merging of Policies

The lower and upper bounds of the corrections are defined by setting parameters d and c_s to $\frac{1}{8}$ and $\frac{1}{2}$ as a fraction of the complete action space (per channel), just as in Figure 3.1. These settings will result in great perturbations on the policy. We refer to Section 3.1 to emphasise that this is indeed desired.

Other hyperparameters are provided in Table 5.1.

Table 5.1: The hyperparameters of Probabilistic Merging of Policies (PMP).

Description	**Symbol**	**Value**
Actor and Critic Network		
Actor learning rate	α_a	$1 \cdot 10^{-4}$
Critic learning rate	α_c	$2 \cdot 10^{-3}$
Buffer size		1e6
Minibatch size	N	64
Discount rate	γ	0.99
Number of heads	K	10
Initial variance (critic)		$3 \cdot 10^{-3}$
Initial variance (actor)		$1 \cdot 10^{-3}$
Target network mixing factor	τ	$3 \cdot 10^{-3}$
Hidden layer sizes		300, 400
OU-Process standard volatility	σ_{OU}	0.3
OU-Process damping	θ_{OU}	0.15
OU-Process timestep	Δt	0.01
Environment		
Maximum length of episode		1000
Maximum number of episodes		200
Action Selection Module		
Resolution (fraction of action space)	d	0.125
Scale factor (fraction of action space)	c_s	0.5
Feedback variance	Σhh	$1 \cdot 10^{-9}$

Table 5.2: The hyperparameters of Predictive Probabilistic Merging of Policies (PPMP).

Description	**Symbol**	**Value**
Neural Networks		
Critic learning rate	α_c	$2\cdot10^{-3}$
Actor learning rate	α_a	$1\cdot10^{-4}$
Predictor learning rate	α_p	$5\cdot10^{-4}$
Buffer size (Actor and Critic)		1e6
Buffer size (Predictor)		1600
Minibatch size	N	64
Discount rate	γ	0.99
Number of heads (actor)	K	10
Initial variance (Critic)		$3\cdot10^{-3}$
Initial variance (Actor)		$1\cdot10^{-3}$
Initial variance (Predictor)		$1\cdot10^{-3}$
Target network mixing factor	τ	$3\cdot10^{-3}$
Hidden layer sizes		300, 400
Other		
OU-Process standard volatility	σ_{OU}	0.3
OU-Process damping	θ_{OU}	0.15
OU-Process timestep	Δt	0.01
Environment		
Maximum length of episode		1000
Maximum number of episodes		200
Selector and Predictor		
Resolution (fraction of action space)	d	0.125
Scale factor (fraction of action space)	c_s	0.5
Prediction variance (fraction of action space)	$\sigma_{\hat{a}_c}$	0.025
Feedback variance	Σ_{hh}	$1\cdot10^{-9}$
Cold-start samples	N_P	1500
Q-filter enabling samples	N_Q	4000

5.1.2. Predicted Probabilistic Merging of Policies

PPMP is implemented as much as PMP in order to allow for a fair comparison. Correspondingly, this section only discusses the differences between the two algorithms. Details of the shared properties are found in the previous subsection.

The predictor module $P(s \mid \phi)$ with parameter set ϕ is implemented in the same way as the policy, as they have a similar function there is no reason to choose a different architecture (except for the number of heads, which is one for the predictor network). The predictor is trained to minimise prediction loss on data from its buffer R_c which contains corrected state-action pairs.

To control the variance in the replay buffer, we set N_P to 1500 samples and $\sigma_{\hat{a}_c}$ to 0.025 as a fraction of the action space (per channel). With the batch size still at 64, this guarantees a significant split of training data recorded at poor performance, a necessity in learning. The Q-filter is enabled after $N_Q = 4000$ samples. Other hyperparameters are found in Table 5.2

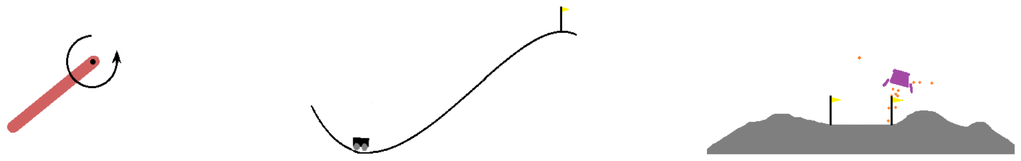

Figure 5.1: From left to right, impressions of the Pendulum, Mountain Car and Lunar Lander environments from the OpenAI gym Brockman et al. (2016). The respective goals in these underactuated problems is to swingup and balance, drive up the mountain and gently land between the flags. The continuous actions are always penalised such that efficiency is incorporated in the objective.

5.2. Benchmark: the OpenAI Gym

The OpenAI Gym, a project by Brockman et al. (2016), allows for standardised comparison between RL algorithms by means of simulated problems. For the algorithms proposed in this work, we use three continuous control environments in 2D.

5.2.1. Underactuated Inverted Pendulum

As a classic example from control theory, the first environment is the inverted pendulum problem, depicted on the left in Figure 5.1. More specifically, we consider the `Pendulum-v0` implementation. From a random initial condition, the goal is to swing the pendulum to its unstable (upright) equilibrium and keep it there. This objective is reflected upon in the reward function by quadratic penalisation of both displacement from the equilibrium (position and velocity) and the applied torque. As a corollary, the return is at most zero, but typically less because of the 'cost to go', i.e. the necesarry control penalty for an optimal transition to the upright equilibrium.

The observations s consist of Cartesian coordinates that define the position of the pendulum's tip along with the angular velocity. There is one action applied, of which the magnitude is contained in a symmetric and bounded interval centered in 0. The maximal torques insufficient to directly steer from the stable to the unstable equilibrium, such that the agents needs to learn a swing-up technique along with the stabilisation. In the Pendulum problem, an episode always lasts 200 timesteps.

5.2.2. Mountain Car

The Mountain Car problem, depicted in the middle of Figure 5.1, comprises a car that needs to build up momentum in order to drive up a hill — again, an underactuated control problem, with a track record in the IRL field (Celemin and Ruiz-del Solar, 2015a; Knox and Stone, 2012). Here, the `MountainCarContinuous-v0` version is considered. The Mountain Car's state space consists of position and speed. The action space is of size one, as the car is controlled by a (reverse) driving force only. When the car reaches the flag, the episode is over and 100 reward points are obtained, discounted by the cumulative squared action. Initialisation is with zero velocity and a random position in the valley.

5.2.3. Lunar Lander

The rightmost impression in Figure 5.1 is of the Lunar Lander problem, where a space pod needs to be flown from the top of the screen to the landing pad situated between the flags. The pod is controlled by means of a main engine (a rocket firing downwards) and two steering rockets on either side. The rockets are only engaged above 50% power, and the two steering rockets are controlled with a single action value to enforce that the signals are perpendicular. In effect, the action space is of size two. The state space is of size eight. It consists of positions, angle, velocities and Boolean values for leg contact. For proper navigation from the top of the screen to the landing pad between

100 and 140 reward units are achieved. Fuel is infinite, but usage is discounted. Gently coming to rest adds 100 points whereas a crash costs 100. In either case, the episode is over.

5.3. Synthesised Feedback

As stipulated in Section 2.4.3, human feedback raises non-reproducible inconsistencies (and training effects) which hinder effective comparability across implementations. As is commonplace in the field of IRL (Knox and Stone, 2012; Griffith et al., 2013; Christiano et al., 2017; Pinsler et al., 2018; Celemin et al., 2018), we employ synthesised feedback to eliminate these distortions from the eventual comparison.

The artificial corrections are obtained by comparison against a reference policy (or 'oracle') and regarding the difference. If the difference exceeds a threshold, a correction is given in the direction of the oracle's policy. For our algorithms, we set this threshold equal to the assumed feedback precision d. For the experiments with `DCOACH`, the performance would be unfairly hindered by applying an assumption that the authors do not make, so the threshold is omitted. To control the feedback rate, a feedback probability is annealed. The exact amount of feedback however depends on the abilities of the agent, and in case all actions are within distance d of the reference policy, no feedback is given regardless the feedback probability. The error rates of the erroneous feedback cases were implemented as the chance that a correction has opposite sign, analogue to a human that presses the wrong button. This approach is adopted from Celemin et al. (2018).

5.4. Human Feedback

In order to test the applicability to actual human feedback scenarios, four tests are conducted with three participants each. Each test (`DCOACH` versus `PPMP`, both in the Pendulum and Mountain Car domain) is done four times with every participants, so both comparisons consist of 24 runs. In each run, one was asked to provide feedback for as long as they deemed it relevant, and in case runs did not converge commit for at least ten minutes before giving up. These ten minutes amount to 60 episodes with feedback in Pendulum. For Mountain Car the amount of episodes within this timespan depends on the performance, but it was often comparable. To record the performance in the absence of feedback, all runs were continued up to 200 episodes.

The participants in these experiments were under 30 years old and of various background. A priori they were given an explanation of the environment, the goal and the interface (keyboard arrows), and we explained that they were to correct an entity that learns to solve the problem. After one dummy run to familiarise participants with their task, the data was gathered in random fashion. As each comparison takes between one and two hours per person, one was given the possibility to pause an experiment when desired.

5.5. Conclusion

This chapter has provided the details on the implementation and testing of this work. First, the implementation of the neural network architectures was covered, for each component in `PMP` and `PPMP` respectively. Along with an account of the hyperparameters used for these existing approaches, the few additional hyperparameters that our methods introduce were motivated. Then, the intended benchmark environments Pendulum, Mountain Car and Lunar Lander were introduced. This was followed by a discussion on reproducible testing by means of synthesised feedback. In addition, the protocol for testing with real human feedback was provided.

Now that the details of implementation and validation have been established, we proceed this writing by presenting the experimental results in the next chapter.

6 Results

The algorithms PMP and PPMP that were presented in Chapters 3 and 4 respectively are evaluated here. The criteria that this evaluation concentrates on are fourfold. Since the incentive of this study is to combine the benefits of human intuition with machine precision and speed, the foremost objective is to have both sample efficiency and final performance. Further criteria are feedback efficiency (minimal user effort) and robustness to erroneous feedback to comply with the imperfections that stem from human factors.

The first reference implementation is DDPG, an instance of deep RL (Section 2.3). The second baseline is DCOACH (Pérez Dattari et al., 2018) DCOACH learns from binary corrective feedback only. For reproducibility, the different algorithms are evaluated for the same random seeds in order to fix the environmental stochasticity. The ten seeds were set by the least significant digits of a nanosecond timer in the first run, to randomise the seeding and avoid seeding bias. In the displayed results, quantities are averages over these random seeds. To improve readability, the results are furthermore smoothed by a moving average filter with window size 5. The depicted hue reflects the standard deviation in the results, but its value is not to be confused with the actual standard deviation at a certain point because of two reasons. First, the smoothing affects the magnitude of the shaded area. Second, the displayed deviation is a bootstrapped estimate of the true standard deviation. Nevertheless, the shaded area allows us to make inferences about the actual variance and this is consistent throughout this writing.

The performance of the oracle will be displayed as a constant line. Although its policy is indeed deterministic and not subject to change, the actual return per episode depends on the initial conditions and does vary. This is expressed by the shaded area.

The sequel of this chapter starts with the simulated results for the Pendulum problem (Section 6.1), followed by those for the Mountain Car (Section 6.2). This includes studies with erroneous feedback. Section 6.3 presents results for the same two environments but then for actual human feedback. In Section 6.4 we present results for the Lunar Lander environment, where we demonstrate that PPMP is able to outperform the demonstrator policy. Thereafter, this chapter concludes in Section 6.5.

6.1. Underactuated Inverted Pendulum, Simulated Feedback

Figure 6.1 contains the results for all four implementations, along with an indication of the oracles performance (purple). The DCOACH (green) study does show some progress but as the feedback is cut off after 100 episodes there is no further learning. If we consider DDPG (red), there is a decent learning curve, however the sample efficiency leaves room for improvement and is moreover weakened by a long period of performance below -1000, which is the worst across all methods.

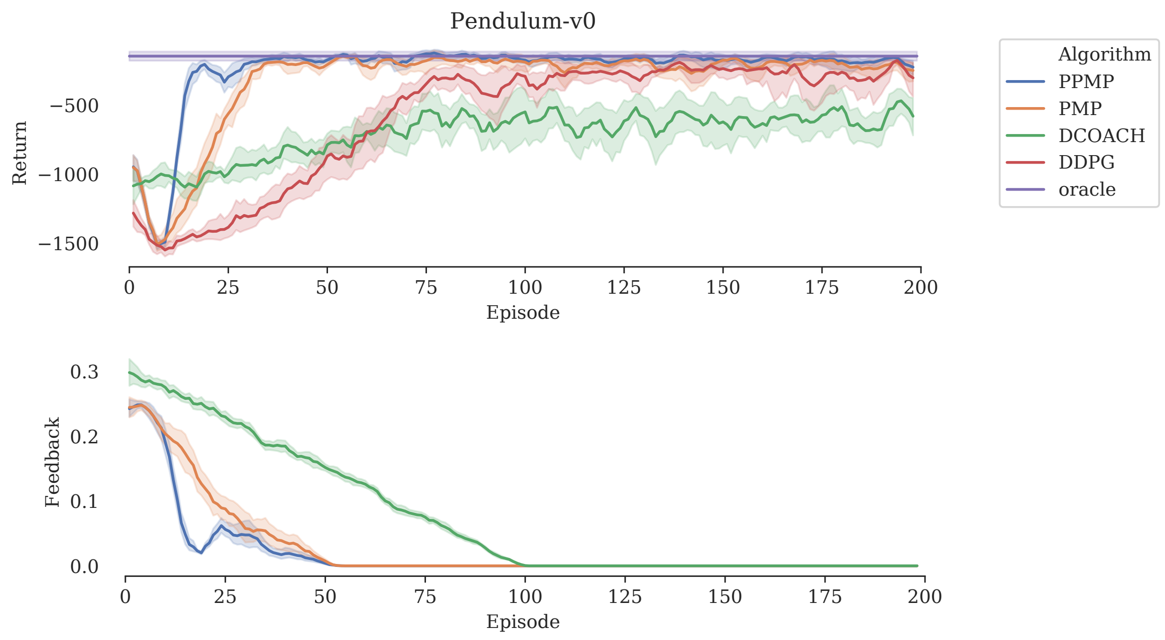

Figure 6.1: The average return and feedback rate in the Pendulum domain. Here, performance is studied with consistent synthetic feedback from an 'oracle'.

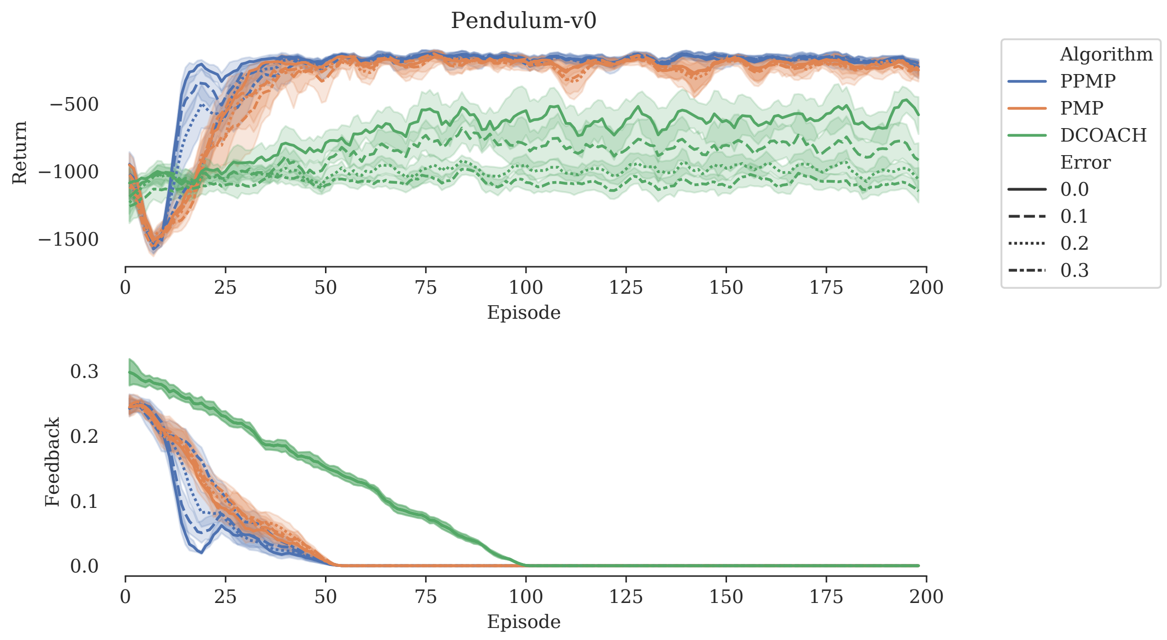

Figure 6.2: The average return and feedback rate in the Pendulum domain. The applied feedback is artificially distorted to assess the robustness of the implementations. The erroneous feedback has greatest adverse effect in methods that rely on a learned model of the feedback, such as `DCOACH` and `PPMP` in its early learning phase.

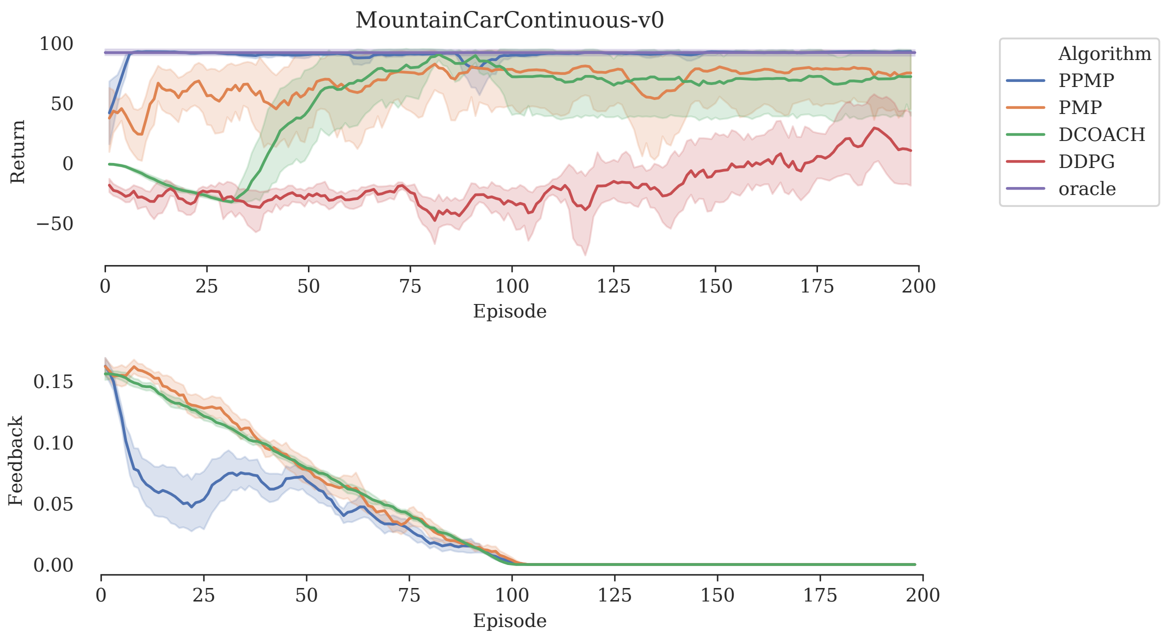

Figure 6.3: The average return and feedback rate in the Mountain Car domain, studied with consistent synthetic feedback.

In the performance curve of PPMP one can see which elements are commanding at a certain time. In the first stage, the performance resembles that of DDPG, but with slight improvements because of the applied corrections. After approximately 10 episodes the predictor enters into force and this gives great performance. It is slightly suboptimal, as is expected by the coarseness of approximation in the predictor module. Slightly later, after approximately 20 episodes, the Q-filter is enabled. As the RL learning process is now in full swing, the actor and critic will sometimes be wrong still. Indeed there is a slight setback in performance and a little increase of resulting corrective feedback, but the performance is largely retained and already comparable with the converged baselines. After some 50 episodes, the RL functionalities have converged to final performance which exceeds the other methods and has less variance. The required feedback is not directly comparable to DCOACH, because the performance is different and our methods only had feedback on the greatest errors. The depicted amount is however far less than in DCOACH.

Figure 6.2 shows the performance of the interactive methods when these are subjected to erroneous feedback. As expected, the DCOACH implementation suffers from the modelling errors caused by the wrong feedback. The performance is brought down over the complete learning trajectory: convergence is slowed down and final performance worse. In PPMP however, the misguidance is only of adverse effect at the time the predictor is in full control (i.e. approximately between 10 and 20 episodes). After 20 episodes, the effects are hardly significant. After 50 episodes there is no apparent deviation from optimality. Being completely model-free, PMP has the greatest robustness. The feedback curves roughly equal those presented in Figure 6.1. As expected by the slightly lesser performance, the erroneous runs require a bit more feedback.

6.2. Mountain Car, Simulated Feedback

In contrast to the Pendulum environment, DDPG now does not learn to solve the problem in the designated time (Figure 6.3) whereas DCOACH is more successful. Nevertheless, there is quite a long learning time in DCOACH and still more feedback would be required before all instances converge (we

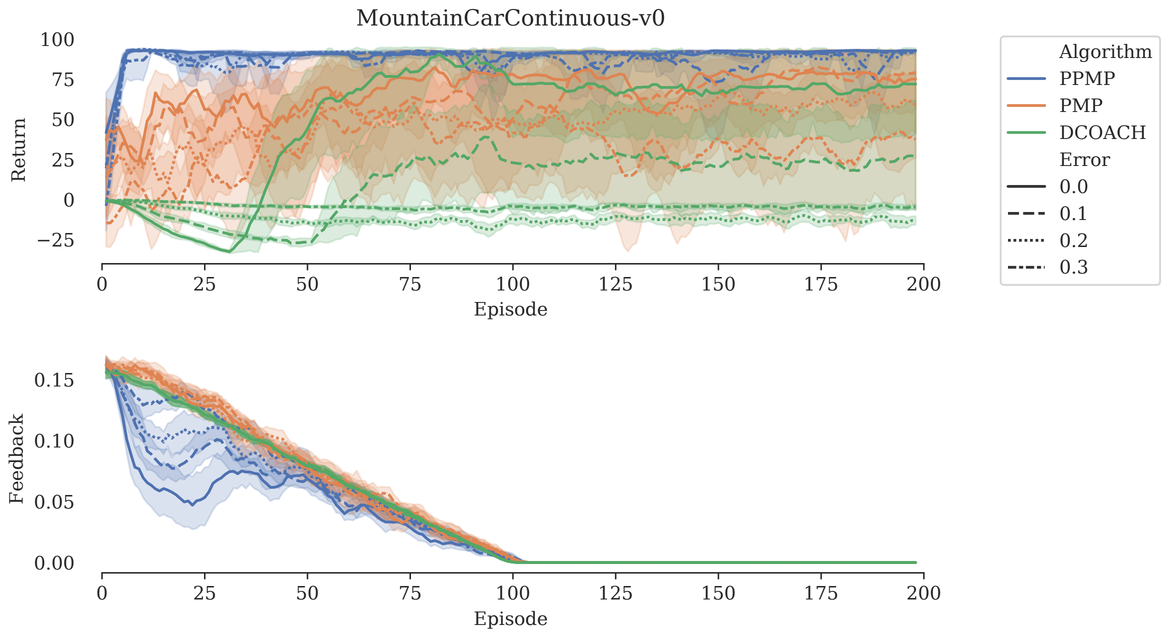

Figure 6.4: The average return and feedback rate in the Mountain Car domain. The applied feedback is artificially distorted to assess the robustness of the implementations.

see the average of methods that either reach or do not reach the flag). PPMP performs up to reaching the flag in a few episodes, and some runs reach the flag even in the first effort. The RL functionalities nevertheless still need a reasonable amount of time to learn the problem, as feedback is given during the first 100 episodes to retain performance. The small dip around the 90th episode indicates that this feedback is not superfluous and the algorithm has not converged yet.

In this problem, the performance of PMP is more behind PPMP than it was in the Pendulum problem. From the shaded area we conclude that most runs nevertheless solve quite soon, but few others never converge such that the average final performance stays behind — this in spite of the significantly greater amount of feedback that PMP receives. Again, the predictor module proves to be useful, as it provides for faster and better performance for lesser feedback. Moreover, it may further help to accomplish persistent exploration, that is here a necessity to gain momentum.

Figure 6.4 illustrates the sensitivity of the interactive methods to imperfect feedback. For DCOACH, almost all runs that were subjected to erroneous feedback failed to reach the flag. PMP proves to be very robust, only the severe case of 30% causes a significant set-back in performance. PPMP handles the imperfections best. Neither the sample efficiency nor the final performance is substantially affected.

Let us now proceed to the next section, where it is reviewed how these results carry over to actual human feedback.

6.3. Real Human Feedback

In this section, PPMP is compared to DCOACH as it was in the previous two sections concerning erroneous feedback, but then with feedback from human participants. The curves presented here are averages of 12 experiments, of which individual plots are found in Appendix B.

Figure 6.5 contains the results for the Pendulum problem. The performance of PPMP is good, and approximately equal to our findings with the simulated feedback. The DCOACH algorithm fails except for one run (Figure B.2), despite a considerable amount of feedback. This is in line with what

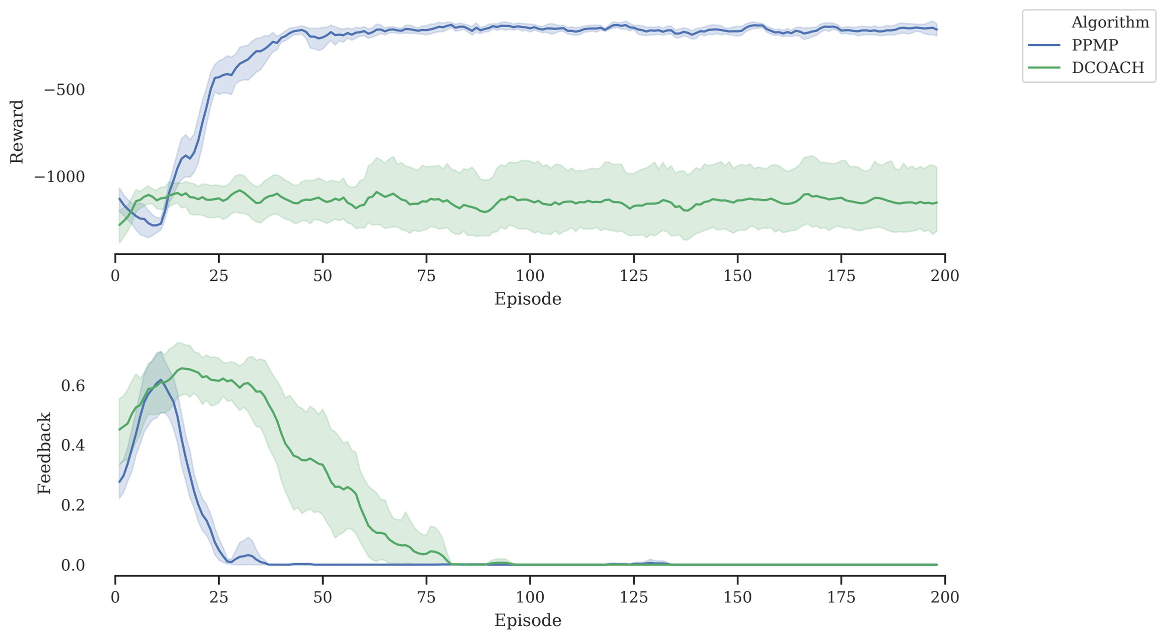

Figure 6.5: The learning curves with actual human feedback in the Pendulum domain.

may be expected from the simulated results.

The main difference with the results from simulation is the feedback rate, which is greater here although the performance is comparable. It reveals that human feedback is indeed not perfect, and there is a discrepancy with simulation because of delays in the feedback (amongst other things, see Section 2.4.3). It could besides be a result of feedback bursts when people hold a key, such that the feedback rate rapidly increases whereas the information gain is not necessarily proportional. Furthermore, people perhaps compensate for earlier erroneous feedback. It could also be that the feedback is simply superfluous. The duration of the feedback here is comparable with the duration of the feedback in the simulated results.

Figure 6.6 concerns the Mountain Car domain. It was previously shown that RL has more difficulties there than in Pendulum. Likewise, PPMP requires little but continuous feedback in the first 100 episodes to retain performance. In these results the performance of PPMP again quickly rises to the maximum. However, participants than presumably think that the algorithm has converged and their task is done. Indeed, after 25 episodes the provided feedback quickly diminishes. The consecutive set-back in performance reveals that the algorithm did however not yet converge. It then takes some episodes to recover, and after that the problem is quite consistently solved.

For DCOACH the performance has a great variance and reflects on all situations. There are runs that solve the problem and from then on always reach the flag with minimal effort and maximum return (Figure B.4), whereas on the other hand there is one that was badly stuck in driving to the left (minimum return). Most evaluations are somewhere in between and only occasionally have a return over 90. These results are in line with our findings in the previous sections where feedback was simulated. It was furthermore observed that the DCOACH solutions tend to drive back and forth a lot, which is neither necessary nor optimal.

6.4. Outperforming the Oracle (Lunar Lander)

This section is not concerned with benchmarking and validation, but demonstrates a typical use case where the feedback has limited performance and is not able to fully solve the problem itself.

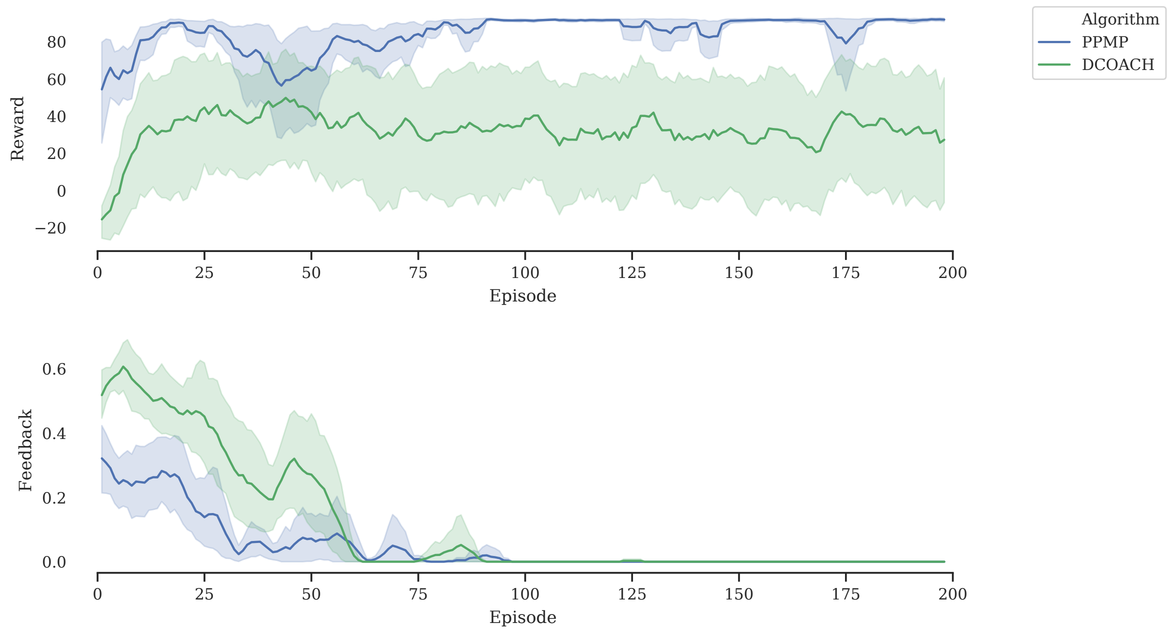

Figure 6.6: The learning curves with actual human feedback in the Mountain Car domain.

This scenario is different from the previous studies with erroneous feedback. Before, incidental mistakes may have been overcome by low-pass dynamics and generalisation, which is possible when the corrections eventually still extend to the end-goal. Here, we generate feedback with a partial policy to that has learned to fly but not to land. It will thus help in learning to fly, but solving the sequel of the problem is left to the agent.

The result of this study is depicted in Figure 6.7, along with the performance of the 'partial' oracle and an evaluation of DDPG. Again, the results are averaged over ten runs. At first sight, the result may not seem very telling, but it is emphasised that the reward function of this environment stresses stable operation by assigning great negative reward to crashes. Only the last 100 reward units that our method obtains correspond to having learned to land properly. As such, our method allows to solve a problem that is otherwise not feasible.

In addition to the results discussed in this chapter, Appendix A discusses supplementary results on hyperparameter stability and scalability of the multihead actor network.

6.5. Conclusion

This chapter discussed the results of our methods in comparison to DDPG and DCOACH. Most results were obtained in the Pendulum and Mountain Car continuous control problems of the OpenAI gym. First, measurements were taken by means of synthesised feedback, in order not to disturb the results by human factors. The main criterion was sample efficiency, and it was demonstrated how our methods have great, sometimes decisive impact in that respect. Further care went out for feedback efficiency, final performance and the robustness to erroneous feedback. Again, PMP and PPMP compared favourably. Next, the applicability of our work to actual human feedback was validated by means of a test with different participants for the same two environments. The results were in line with those from simulated feedback. In addition it was demonstrated how PPMP solves a problem (Lunar Lander) even when the teacher has limited knowledge. As desired, the sample efficiency of DDPG is drastically improved by our methods, and the poor performance stage is in some cases almost completely evaded. In the next and final chapter this work will be concluded.

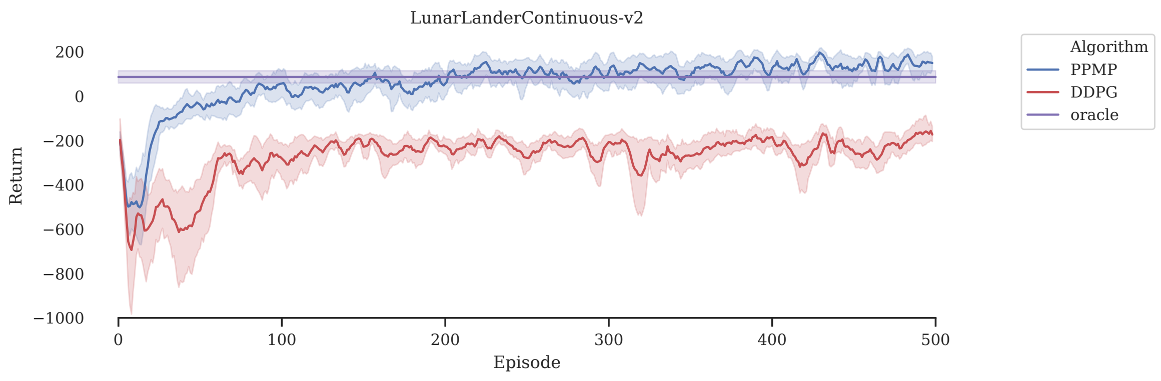

Figure 6.7: In the case that the feedback source has limited performance, the partial guidance still accelerates the learning. Subsequently, the autonomous learning abilities of `PPMP` solve the problem. Note that the last 100 reward units are obtained for gently coming to rest between the flags, whereas the oracle had only learned to fly and not crash.

7 Conclusion

Deep Reinforcement Learning is a promising way to solve high-dimensional control and decision-making problems, as it autonomous and does not require modelling. A disadvantage is however the excessive need for interaction between the agent and the problem, which limits applicability to real-world scenarios. In this work, it is discussed how binary corrective feedback may be used as a probabilistic exploration signal. By slight modification of the DDPG algorithm, an uncertainty of the policy was obtained and coupled with the magnitude of the provided corrections. As such, we established effective exploration that is initially very expressive, but more subtle upon convergence of the policy to facilitate refinements. This algorithm was named Probabilistic Merging of Policies (PMP).

In a second, extended algorithm named Predictive Probabilistic Merging of Policies (PPMP), a predictor network provides for estimates of the corrected policy which can substitute for the actual policy when it increases the value of the state-action pair (especially during early learning). The predictor module helps to generalise the corrections and improve memorisation, such that there is a lesser need for feedback and greater, more consistent final performance. However, a naive implementation of the predictor makes DDPG (even more) prone to instability. We have proposed three solutions to overcome this problem, which require a minimal amount of hyperparameters, and furthermore argued that there is a wide range in which these parameters are successfully set.

In effect, our methods are easily implemented and make realistic assumptions about the feedback. It does not need to be a full demonstration, expertise may be limited, we do not assume feedback is abundant, neither is simulation or post-processing required. Nevertheless, both our algorithms PMP and PPMP consistently improve on sample efficiency, final performance and robustness in a comparison against pure RL (DDPG) and learning from corrections only (DCOACH). It was furthermore demonstrated that our methods require little feedback and are able to outperform the demonstrator.

In conclusion, we have established the desired improvement of sample efficiency by using corrective feedback as probabilistic exploration. The applicability of deep reinforcement learning methods for continuous control is thereby extended.

Further Research

Let us start this discussion with suggested improvements ($\diamond$) and subsequently reason about possible extensions ($*$) to this study.

- $\diamond$ The idea that the feedback should be rough perturbation of the policy has led to promising results in simulated experiments. However, it may cause for unwanted side-effects when applied to physical systems, and perhaps damage. We believe further work on the *Selector* module (for example in the direction of low-pass filtering) is essential before our algorithms are safely applied to arbitrary physical systems.

- $\diamond$ We have used the multihead actor network to estimate the covariance in the policy. The subsequent application in the Selector worked out, but we have not scrutinised this estimation itself, and we are interested to see how well the estimates actually reflect the abilities of the actor. How the covariance relates to state visitation, generalisation, and optimality is not clear, neither do we know what factors possibly bias this estimation. In the words of Osband et al. (2016): *"... this area of uncertainty estimates for neural networks remains an important area of research in its own right".*

- $*$ From the possible extensions to this study, an exiting avenue would be to refine the probabilistic part. In particular, we would like to either extend the Gaussian approximation using mixture models (Haarnoja et al., 2018) or dispose of it and connect with full distributional learning (Bellemare et al., 2017; Barth-Maron et al., 2018), possibly combined with uncertainty-handling estimation of human feedback (Wout et al., 2019). It would presumably yield better estimates of the current abilities and allow for more sophisticated action selection, for example by means of posterior or Thompson sampling (Osband et al., 2016).

- $*$ The proposed methods are naturally compatible with continuous feedback, such as from a joystick, or feedback with a extended discretisation (e.g. $h \in (-5, -1, 0, 1, 5)$ to let the user affect the magnitude of correction). It would be interesting to study if one of these could further improve learning efficiency.

- $*$ A more sophisticated re-sampling of past experience (de Bruin et al., 2018), that possibly distinguishes samples that have been corrected, could improve the learning of the actor and the critic. In addition, the predictor's training batch could perhaps be appended with the latest correction (at most once, or a few times) in order to account for temporal correlations in the feedback.

- $*$ To account for delays between intended human corrections and actual processing, it could be desirable to adopt a *credit assigner* module that distributes feedback over a range of samples (Knox and Stone, 2009; Celemin and Ruiz-del Solar, 2015a). Although it would be hard to incorporate credit assignment directly in the learning of the problem, there are interesting possibilities for this in the learning of the predictor.

A Additional Results

Here, in the first appendix, we present additional studies as further support to the main matter. Section A.1 presents an assessment of the hyperparameter sensitivity. In Section A.2 it is studies how multihead networks scale.

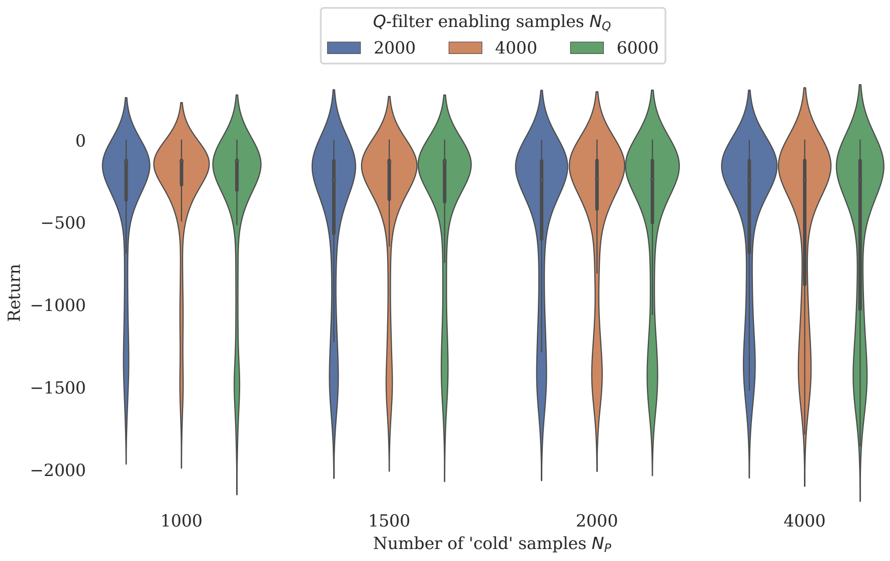

Figure A.1: A fitted density plot of the average return in the first 75 episodes of the Pendulum environment. Each hyperparameter set is evaluated five times. N_Q and N_P do not have a great impact on the stability and convergence properties, but finding optimal values may lead to a slightly faster learning curve.

A.1. Hyperparameter Sensitivity

Some deep reinforcement learning approaches are notoriously brittle (Duan et al., 2016; Henderson et al., 2017), and this is certainly the case for DDPG (Lillicrap et al., 2016). We have yet covered the need for sample efficiency and and improved on it, but meanwhile hyperparameter sensitivity is a complementary aspect that should not be overlooked. If a method requires extensive hyperparameter searches, the saving on required interaction can never materialise. In this study the brittleness of DDPG is out of our scope, as we deem our method applicable to more stable off-policy approaches (e.g. Haarnoja et al., 2018) as well. Nonetheless, we will assess the sensitivity of the newly introduced parameters of the predictor module.

In Section 4, where the design of PPMP was presented, it was stated that the two additional hyperparameters N_Q and N_P do not require meticulous tuning since there is a wide range in which these are successfully set. In order to motivate the alleged robustness, we here present twelve permutations of the hyperparameter subspace, each evaluated five times in the Pendulum domain for 15000 timesteps (75 episodes).

From the distributions in Figure A.1 we may conclude that these parameters slightly effect the cumulative return, but there is indeed no brittleness as all tests were successful.

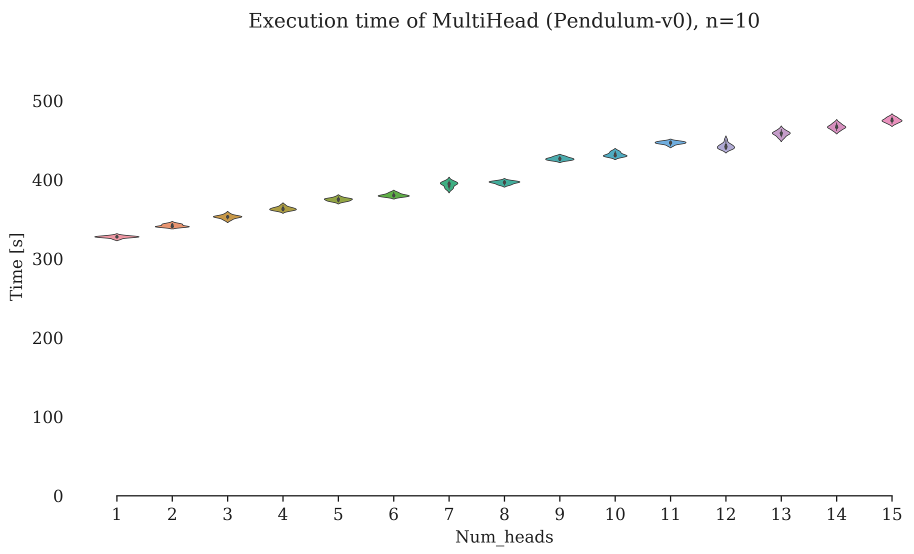

Figure A.2: For a given problem and amount of samples, the execution time of our multihead actor implementation scales approximately linear in the amount of heads K.

A.2. Execution Time vs. Number of Heads

As this works aims to improve on the feasibility of model-free learning control, we here address a secondary concern that may arise upon application: the computational complexity. Most modifications and parameters that we suggest add a fixed amount of operations to the RL baseline. Changing the amounts of heads in the multihead actor however does affect the computational complexity, and in this appendix we will infer the scaling properties of this parameter.

These results were collected for ten full runs (200 episodes) in the Pendulum domain using a Asus® UX490 machine running Linux Ubuntu 18.10. This is not a dedicated platform, but background processes were kept at a minimum. The architecture and hyperparameters are as presented in Chapter 5, implemented in Python as available at github.com/janscholten/ppmp.

In figure Figure A.2 the scaling of the multihead implementation is depicted, up to fifteen heads. The execution times (obtained with the bash command `time`) are narrowly distributed. As should be expected, the execution time scales linearly with the additional parameters, save a little kink between 7 and 12 heads — presumably a memory bottleneck. From this result, we may conclude that the multihead approach does not conflict with feasibility as the additional cost is very manageable.

B
Human Feedback: Individual Results

In this appendix the results obtained with actual human participant are presented individually, for each algorithm and for each environment.

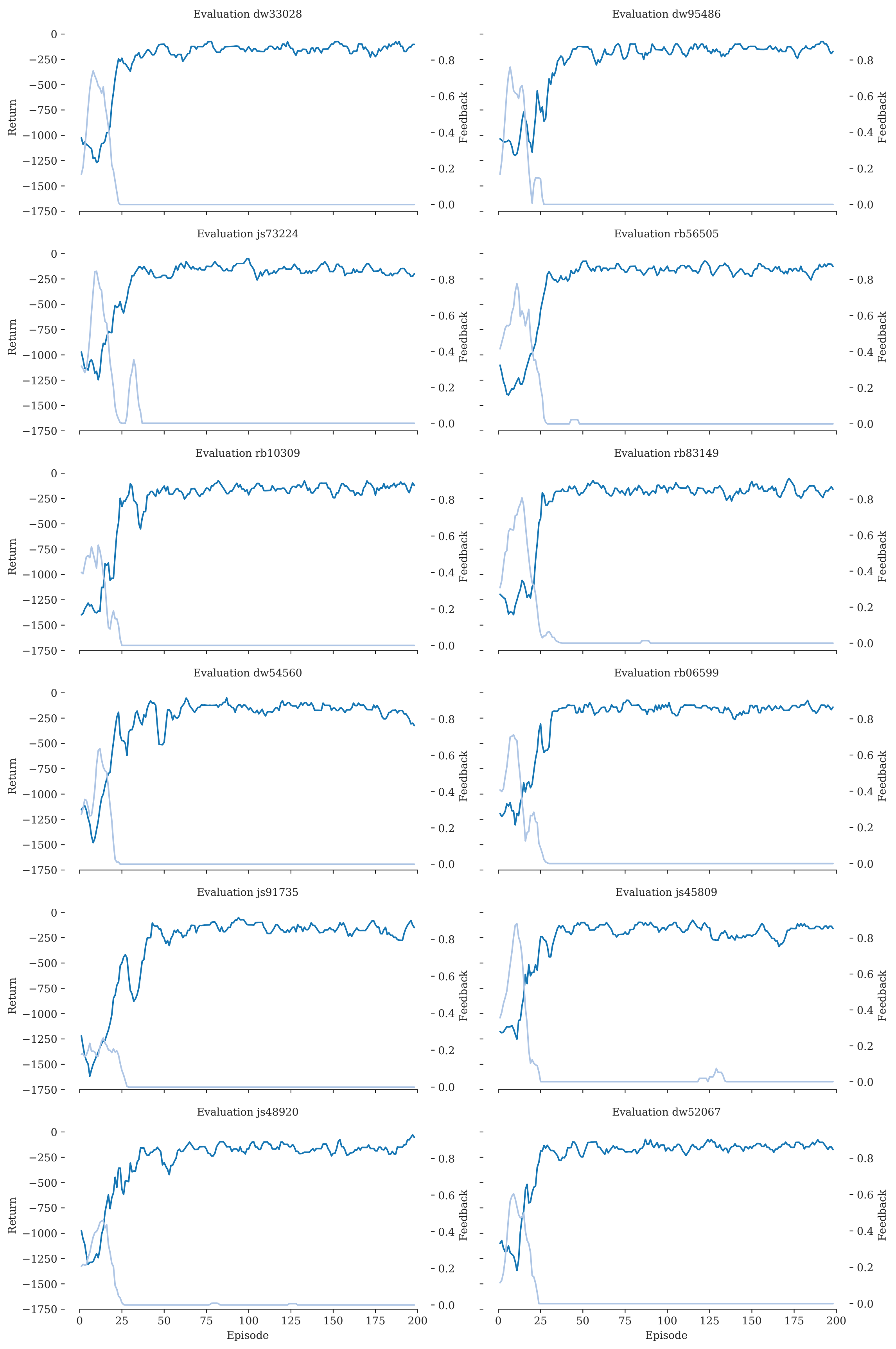

Figure B.1: Experiments with real human feedback for PPMP in the Pendulum domain.

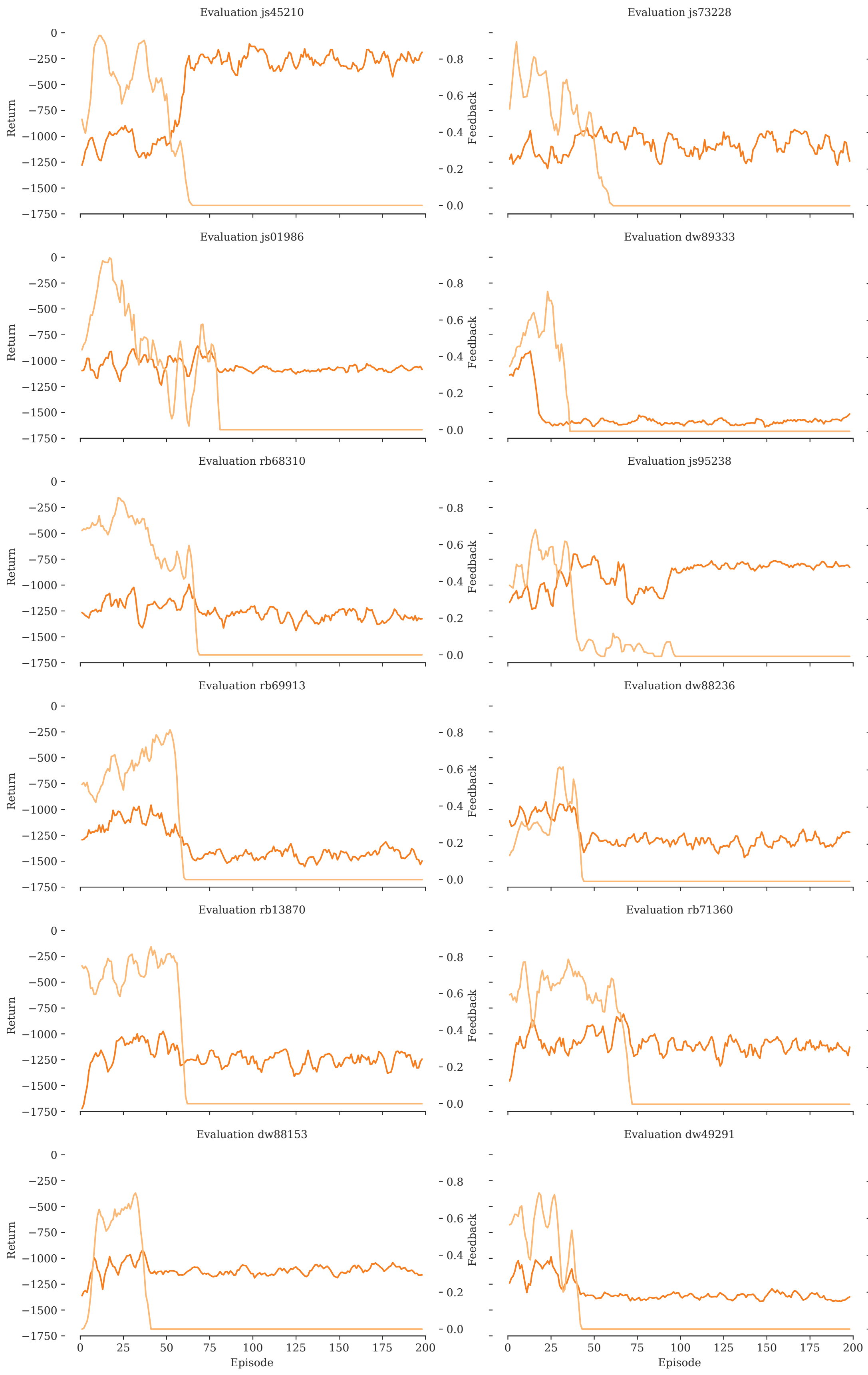

Figure B.2: Experiments with real human feedback for `DCOACH` in the Pendulum domain.

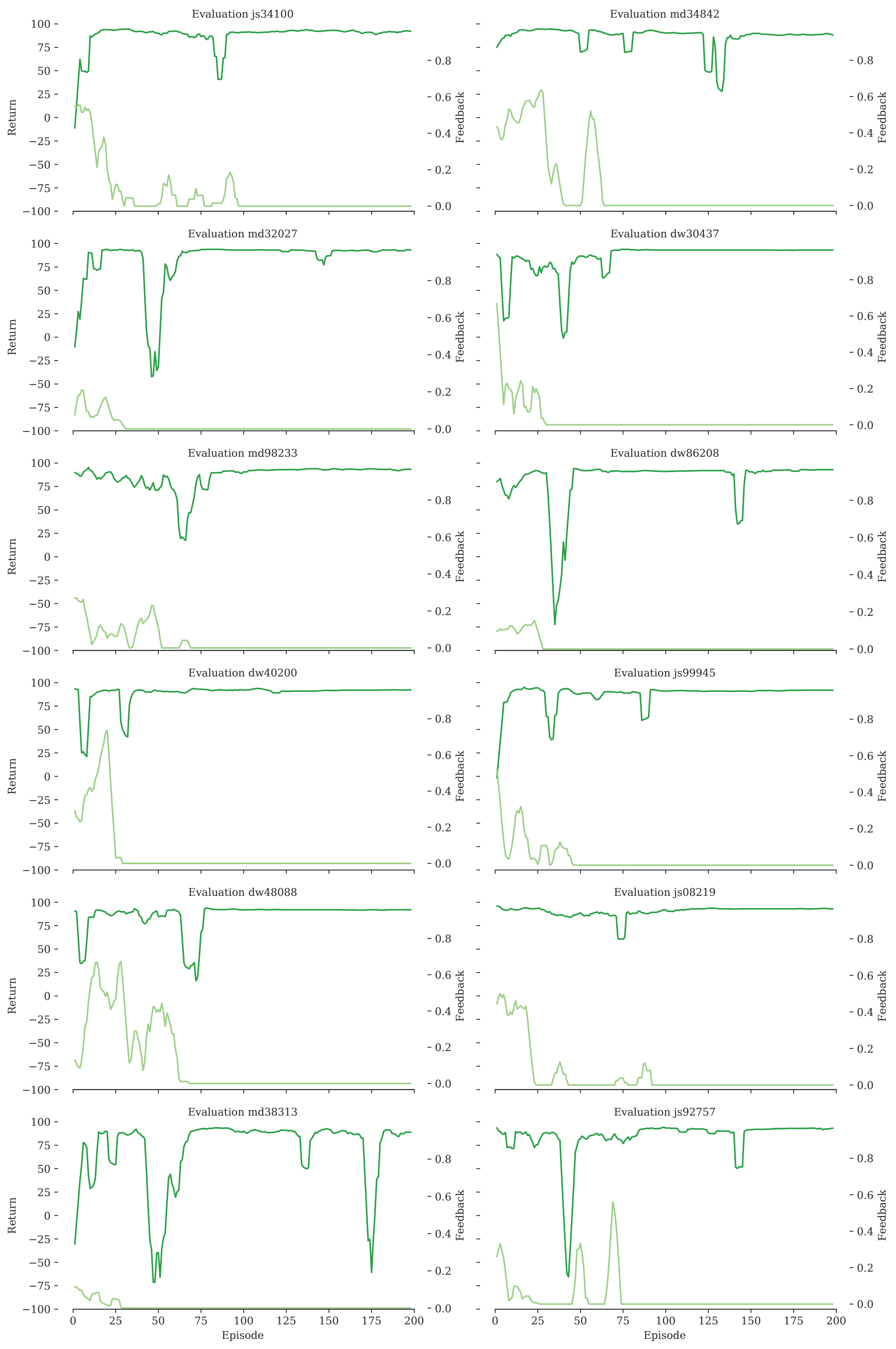

Figure B.3: Experiments with real human feedback for PPMP in the Mountain Car domain.

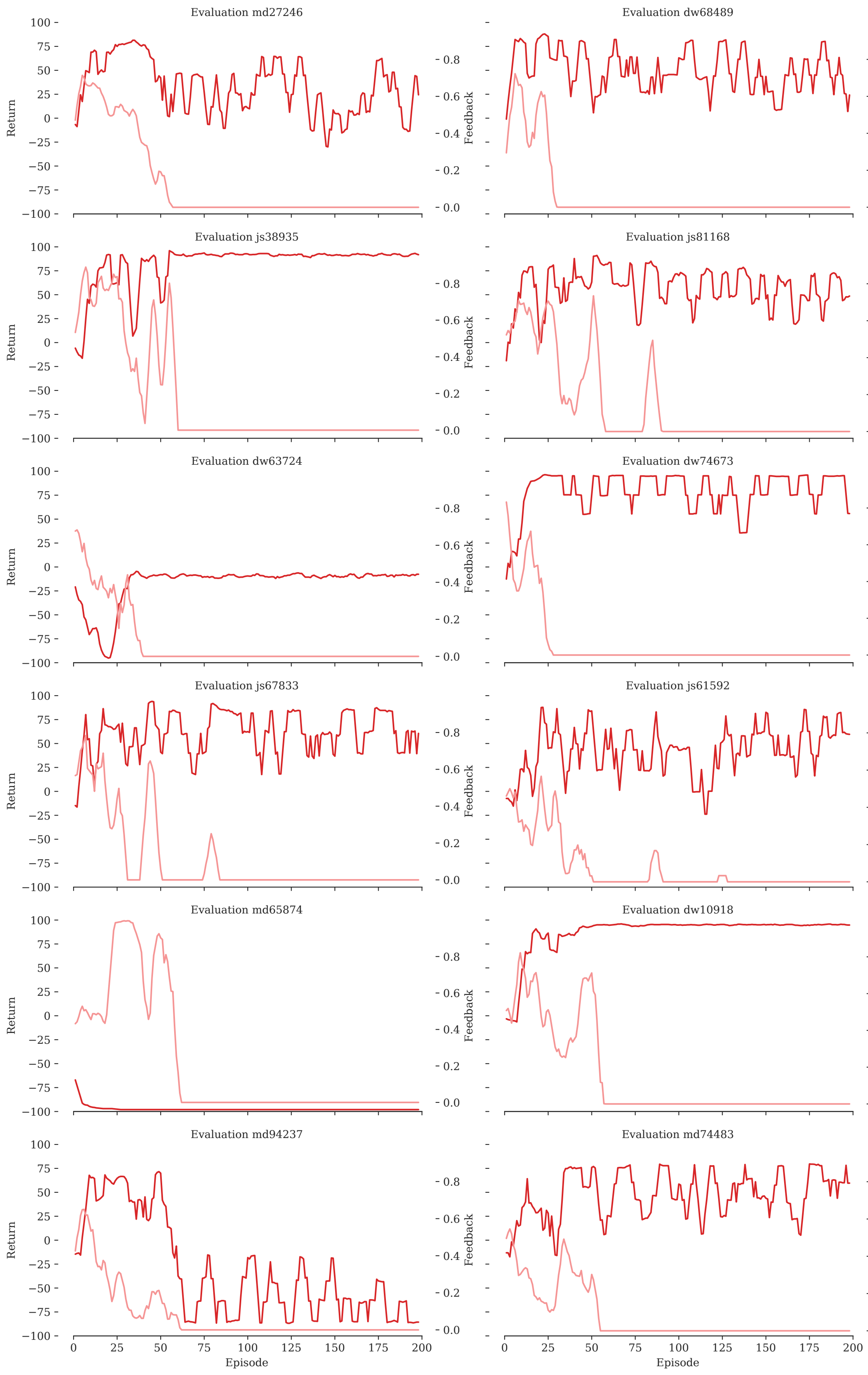

Figure B.4: Experiments with real human feedback for `DCOACH` in the Mountain Car domain.

C
Paper

A compact treatise of this work is available on ArXiv.com (Scholten et al., 2019) and submitted to the 58th Conference on Decision and Control (currently under review). The main difference with this writing is that only Predictive Probabilistic Merging of Policies (`PPMP`) is explicitly presented, and Probabilistic Merging of Policies (`PMP`) is only included as an ablation.

Deep Reinforcement Learning with Feedback-based Exploration

Jan Scholten, Daan Wout, Carlos Celemin, and Jens Kober

***Abstract*— Deep Reinforcement Learning has enabled the control of increasingly complex and high-dimensional problems. However, the need of vast amounts of data before reasonable performance is attained prevents its widespread application. We employ binary corrective feedback as a general and intuitive manner to incorporate human intuition and domain knowledge in model-free machine learning. The uncertainty in the policy and the corrective feedback is combined directly in the action space as probabilistic conditional exploration. As a result, the greatest part of the otherwise ignorant learning process can be avoided. We demonstrate the proposed method, Predictive Probabilistic Merging of Policies (PPMP), in combination with DDPG. In experiments on continuous control problems of the OpenAI Gym, we achieve drastic improvements in sample efficiency, final performance, and robustness to erroneous feedback, both for human and synthetic feedback. Additionally, we show solutions beyond the demonstrated knowledge.**

I. Introduction

Contemporary control engineering is adopting the data-driven domain where high-dimensional problems of increasing complexity are solved, even if these are intractable from a classic control perspective. Learning algorithms, in particular Reinforcement Learning [1], already enable innovations in robotic, automotive and logistic applications [2]–[4] and are on the verge of broad application now that data becomes ubiquitous [5]. There are many applications also beyond the classical control engineering domain, such as HIV [6] and cancer treatment schedules [7]. A possibly extension to diabetes treatment could have great impact [8]. In contrast to model-based control, RL is able to retain optimality even in a varying environment, and modelling of dynamics or control design is not needed.

This study concerns deep RL (DRL), the leading approach for high-dimensional problems that uses neural networks to generalise from observations to actions. DRL can greatly outperform humans [9] in virtue of machine precision and reaction time. However, DRL requires extensive interaction with the problem before achieving final performance. For real-world systems that have restrictions on interaction, the sample efficiency can be decisive for the feasibility of the intended application [10]. Improving sample efficiency is thus essential to the development of DRL and its applications.

In contrast to autonomous learning algorithms, humans are very effective in identifying strategies when faced with new problems. In many cases we achieve decent performance in the first try, despite poorer precision and reaction time that limit final performance. Indeed, from the sample efficiency

All authors are with Cognitive Robotics Department, Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, The Netherlands and reachable via jan@jjscholten.com

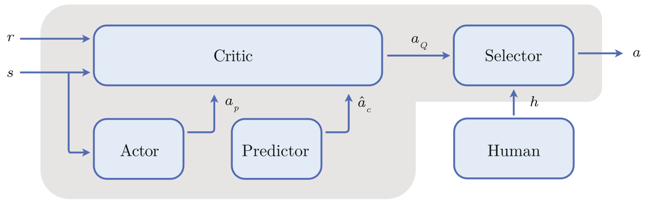

Fig. 1. In the suggested approach, human feedback is combined either with the policy or a prediction of the corrected action. The critic discriminates these with respect to the estimated value of the action. The magnitude of the correction is estimated in the *Selector* and proportional to the estimation variance of the policy. The elements in the grey area constitute an autonomous learner.

perspective, human and RL performance are complementary and incorporating human insight into a learning algorithm is a great way to accelerate it.

Some existing RL methods update the policy with additional human feedback, which is provided occasionally [11], [12]. In contrast, we propose to keep the original RL process and use the human feedback for exploration, as Nair et al. do with demonstrations [13]. Focussing on DRL, there are methods that learn from a priori demonstrations [14] or intermittently collect those (dataset aggregation) [15]. In contrast to corrective feedback, demonstrations are not always available or even possible (there may be limitations in the interface or expertise), besides that they may require manual processing [14] or simulation [13]. Likewise, methods that receive preferences between trajectories can be powerful [16] but they assume the availability of simulation, which is not generally realistic. As a general measure, there is evaluative feedback [2] but we believe that there currently is no method that uses (binary) corrective feedback [12] to accelerate DRL. Yet for the purpose of conditioning the exploration of continuous control problems this would be a natural choice. Moreover, corrective feedback promises to be more effective than evaluative rewards especially in larger action spaces [17].

We present a pioneering combination of DRL with corrective human feedback for exploration, to efficiently solve continuous control problems (Fig. 1). We revisit the question of how the current estimate of the policy is best combined with feedback, and subsequently derive a probabilistic algorithm named Predictive Probabilistic Merging of Policies (PPMP) that improves the state-of-the art in sample efficiency, is robust to erroneous feedback, and feedback efficient. Whilst the proposed assumptions remain realistic, the introduced techniques are moreover generic and should apply to many deep actor-critic (off-policy) methods in the field.

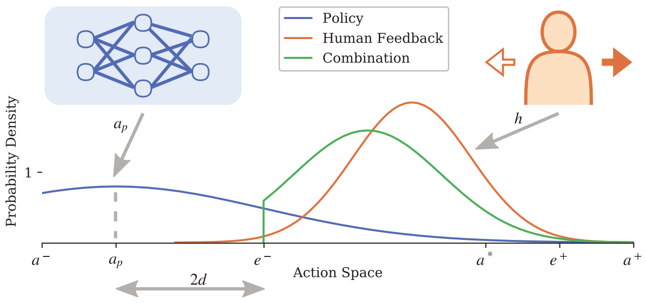

Fig. 2. Using respective covariances, the policy is combined with human feedback in the action space. The resulting distribution on the action that is selected, is truncated such that corrections always have significant effect and the given information cannot dissipate in case of an overconfident policy.

Our approach is motivated by four ideas:

Action Selection: After first evidence by Knox & Stone [18], it later were Griffith et al. [19] who made a strong case for how human feedback is given with respect to the action (sequence) and it is most effective to directly adjust the actions when feedback is obtained. Their algorithm outperformed other evaluative feedback methods that instead affect the actions indirectly (modification of the policy).

Significant Error: If we consider the early learning phase, where the policy is useless but the human feedback most valuable, we believe feedback is received in case of a significant error as to help the agent develop a notion of the task rather than to communicate refinements. Indeed, a recent study demonstrated that vigorous initial exploration is beneficial for sample efficiency [20]. Moreover, we argue that the instantaneous precision of human feedback is then rather coarse (in contrast, corrections for steady-state errors of an almost converged policy may be smaller). Accordingly, this limitation is quantified by defining a precision d expressing a region of indifference per dimension.

RL for Fine-stage Optimisation: Reinforcement learning is superior in final performance due to its precision and reaction time [21]. From our point of view, it should therefore be allowed to autonomously optimise during the later learning phases, such that local optima are identified (e.g. using gradients) independent from past feedback.

Probabilistic Approach: Griffith et al. [19] proposed a probabilistic combination of the policy and feedback distributions to determine the action. Because the policy and feedback estimates are balanced by their respective accuracy, such approaches are very effective and robust. In the words of Losey & O'Malley: *'When learning from corrections ... [the agent] should also know what it does not know, and integrate this uncertainty as it makes decisions'* [22]. We subscribe to this point of view and furthermore emphasise past success of using uncertainty in other fields, such as the Kalman filter or localisation algorithms [23]. However, whereas Losey & O'Malley estimate the variance in the corrections [22], we consider the feedback (co)variance fixed (d in *Significant Error*) and argue that the correction size is inversely proportional to the performance of the agent.

It immediately becomes apparent that some of these ideas align, e.g., that corrections are inaccurate (*Significant Error*), but do not need to be accurate, since RL will efficiently identify local optima (*RL for Fine-stage Optimisation*). However, before connecting the dots, let us complete this motivation with the assumption that, given the assumed area of indifference of *Significant Error* expressed by d (Fig. 2), RL is able to identify the local optimum. In other words, *Significant Error* and *RL for Fine-stage Optimisation* concern overlapping regions and the global optimum is attained if the feedback brings us in proximity.

As a corollary of the above statements, we develop a learning method where actions are obtained by significant modification of the policy in direction of the obtained binary feedback. A probabilistic manner that reflects the current abilities of the agent determines the magnitude of correction. This method strongly reduces the need for interaction and furthermore improves final performance. Autonomy and optimality are furthermore preserved, since there will be no feedback when the performance is deemed satisfactory, and our method then resorts to its RL principles.

II. Background

This study is defined in a sequential decision making context, in which the Markov decision process serves as a mathematical framework by defining the quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. This set consists of state-space $\mathcal{S}$, action-space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, and constant discount rate γ [1].

The computational agent interacts with an environment by taking actions a_k (where convenient, we omit the time index k) based on its current state s_k and will then end up in a new state s_{k+1} and receive a reward r_k and human feedback $h_k \in \{-1, 0, 1\}$ to indicate an advice of the direction in the action space, wherein the agent could explore. The objective of the agent is to learn the optimal policy $\pi^*(s)$ that maximises the accumulated discounted reward $R = \sum_i \gamma^i r_i$ And we assume the feedback aligns with this goal. Along with the policy contained in a neural network called the *actor*, the agent will have a network called *critic* which learns to predict the *value* of a state-action pair, i.e., the Q-function $Q(s, a) = \mathbb{E}[R \mid \pi, s_k = s, a_k = a]$. This deep actor-critic approach was introduced in [24]. Their work is the basis of the RL-functionalities used here, such as target networks π' and Q' and replay buffers B, although we consider our scheme could also be applied to other RL algorithms.

III. Predictive Probabilistic Merging of Policies

The aforementioned point of view materialises in our new learning algorithm PPMP, of which we will discuss each element in one of the following subsections.

A. Combining Policy Information in the Selector

For the sake of this explanation, let us temporally assume non-erroneous corrective feedback h on policy a_p that indicates the relative location of optimal action a^* (all scalar, as in Fig. 2). With reference to the *Action Selection* statement, our approach is to immediately alter the actors suggested a_p

in the direction of h, such that the eventually selected action $a = a_p + \hat{e}h$ (the orange distribution in Fig. 2) and $\hat{e}$ being an estimate of the absolute error $|a_p - a^*|$.

Deriving from the Kalman filter, the unknown magnitude of the error is estimated using the covariance of the policy (the prediction) and the feedback (an observation) in a module that we call the *Selector*. It is assumed that the magnitude of the error will diminish over time along with the covariance of the policy $\Sigma_{a_p a_p}$. With the covariance of the feedback Σ_{hh} as a known constant and $\Sigma_{a_p a_p}$ obtained as described in Sec. III-C, let $\hat{e} = G\,\mathrm{diag}(c_s) + \mathbf{1}c_o^T$, (lines 9-11 in Algorithm 1) where the constant vectors c set the bounds on $\hat{e}$ as described in the last two paragraphs of this section, $\mathbf{1} = [1, 1, \ldots, 1]^T$ and $G = \Sigma_{a_p a_p}(\Sigma_{a_p a_p} + \Sigma_{hh})^{-1}$. Note that $G \in (0, 1)$ (all Σ are positive definite by definition) is analogue to the Kalman gain as a dimensionless trade-off measure. When the policy shows large covariance, the corrections will have larger effect and facilitate vigorous exploration. And inversely, corrections will be more subtle upon convergence of the policy. Besides that, the exploration is automatically annealed over time by the decrease of $\Sigma_{a_p a_p}$, its effect is state-dependent and tailored for every action channel, respecting correlations.

The relation of G to $\hat{e}$ (line 11) is defined using two vectors of which the length equals the dimensionality of the action space. First, let us discuss the relevance of offset c_o. With reference to *Significant Error*, a lower bound on e is $e^- = a_p + hd$. Moreover, for the action selection we may further restrict the search by using the fact that a is guaranteed to be closer to a^* than a_p even if $\hat{e} = 2d$. Setting $c_o = 2d$ accordingly sets an effective lower bound on the corrections. Apart from the optimisation perspective, it is always desired to apply a significant correction in case feedback is provided. First, it will avoid frustration of the user, who actually observes the effect of the feedback. Second, the information is not preserved otherwise.

The scale c_s allows us to set an upper bound e^+ for the applied corrections. From the perspective of using human feedback as exploration, let us consider the case where the policy suggests some negative action and receives $h = 1$ since optimality is contained in the positive half of the action space (Fig. 2). Although we cannot make any general statements about the reachability of the state-space, it is clear that feedback can only have the intended effect when $\hat{e}$ is large, else there is no escape from the wrong half of the action space.

B. Integrating the Selector with Autonomous Learning

The ideas established in the previous paragraph raise requirements for the eventual algorithm. The probabilistic combination of the policy and the feedback results in off-policy data in a continuous action space. As critic-only methods are suitable for a discrete action space whilst actor-only methods are on-policy, an off-policy actor-critic scheme remains as the evident choice.

Fig. 1 illustrates how the system is interconnected and the actions selected. It is assumed that the human provides binary feedback signals h occasionally and bases this on the observed state sequence (and possibly the actions). Delays between human perception and feedback are not taken into account. In order to memorise and generalise the advised corrected samples, those corrected actions are estimated in the *predictor*, a supervised learner further discussed in Sec. III-D. First, the Q-filter (critic) decides whether the policy's action a_p or the estimated corrected action $\hat{a}_c$ is preferred as the suggested action a_Q (line 7). Then, in accordance with the description in the previous paragraph, the selector module adjusts a_Q with respect to h and we arrive at the actually executed action a (line 11). In case feedback is not provided the algorithm relies on its own policy, including exploration noise. Autonomy is hereby preserved.

C. Multihead Actor Network

In contrast to DDPG [24] we need not only to estimate an action, but furthermore to estimate the covariance in this estimate. In [25] is established that the uncertainty over a deep neural networks output may be obtained from multiple passes with dropout. However, in the context of RL, [26] reports how a multihead neural network that maintains multiple hypotheses is a more consistent approach to generate posterior samples than dropout. Whereas in their study the eventual purpose of the multihead network is to use the posterior for exploration rather than to quantify confidence (as desired for our approach), [27] indeed establishes how the multiple hypotheses provide for accurate estimation of abilities in a deep learning classification problem. As it is furthermore desired to have efficient and scalable estimation, we apply the multihead architecture as discussed in [26] to the actor network.

Effectively, the modification of a regular actor network to its multihead counterpart results in K copies of the output layer that estimate the optimal action $a_j = \pi_j(s \mid \psi)$ (line 5), where j indicates the head and ψ is the parameter set of the network. For the training, we establish an extension to the sampled policy gradient in [24] that features individual values of $\nabla_a Q$ and $\nabla_\psi \pi$ for each head. This sampled multihead policy gradient is given by

$$\nabla_\psi J^\pi \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q(s, a|\theta)|_{a=\pi(s_i)}\ \nabla_\psi \pi(s|\psi)|_{s=s_i}\ , \quad (1)$$

with a slight abuse of notation in the row-wise expansion of $\nabla_a Q$ that contains evaluations for all K policies in π. To determine the policy during action selection, we choose to randomly select a head j_e per episode, preserving both temporal consistency and compliance with multimodalities (not preserved when averaging). For the training of the critic (line 13 in Algorithm 1), $\pi'_{j_i}(s_{i+1} \mid \psi')$ is evaluated for j_i, the same head as in a_i.

D. Predictor Module

The corrected actions are estimated as $\hat{a}_c = P(s \mid \phi)$, where P is the prediction network with parametrisation ϕ, trained with human-corrected samples (s, a) from buffer B_c. Whilst these predictions can greatly improve the performance

Algorithm 1 Predictive Probabilistic Merging of Policies

```
1:  Initialize:
        Neural network parameters $\theta, \theta', \psi, \psi', \phi$
        Replay buffers $B$ and $B_c$
        Feedback covariance $\Sigma_{hh}$
        Scale $c_s$, and offset $c_o$
2:  for episode $e = 1$ to $M$ do
3:      Initialize:
        Ornstein-Uhlenbeck process $\nu$
        Randomly set active head $j_e$
4:      for timestep $k$ = 1 to $T$ do
5:          $a_p \leftarrow \pi_j(s_k \mid \psi) + \nu_k$
6:          $\hat{a}_c \leftarrow P(s \mid \phi) + \mathcal{N}(0, \sigma_a)$
7:          $a_Q \leftarrow \arg\max_a Q(s, a)|_{s=s_k, a=a_p \vee a=\hat{a}_c}$
8:          if Feedback $h_k$ is given then
9:              $\Sigma_{a_p a_p} \leftarrow \mathrm{cov}(\pi(s_k \mid \psi))$
10:             $G \leftarrow \Sigma_{a_p a_p}(\Sigma_{a_p a_p} + \Sigma_{hh})^{-1}$
11:             $a_k \leftarrow a_Q + (G\,\mathrm{diag}(c_s) + \mathbf{1}c_o^T)h$
12:             Store $(s_k, a_k)$ in $B_c$
13:         end if
14:         Obtain $s_{k+1}$ and $r_k$ by executing $a_k$
15:         Store transition $(s_k, a_k, r_k, s_{k+1}, j_e)$ in $B$
16:         Sample $N$ tuples $(s_i, a_i, r_i, s_{i+1}, j_i)$ from $B$
17:         Compute target $Q$-values
                $y_i = r_i + \gamma Q'\left(s_{i+1}, \pi'_{j_i}(s_{i+1} \mid \psi') \mid \theta'\right)$
18:         Update $\theta$, $J^Q = \frac{1}{N}\sum_{i=1}^N (y_i - Q(s_i, a_i \mid \theta))^2$
19:         Update $\psi$ using multihead policy gradient (1)
20:         Randomly sample $N$ transitions $(s_i, a_i)$ from $B_c$
21:         Update $\phi$, $J^P = \frac{1}{N}\sum_{i=1}^N (P(s_i \mid \phi) - a_i))^2$
22:         Update target network $Q$ : $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
23:         Update target network $\pi$ : $\psi' \leftarrow \tau\psi + (1-\tau)\psi'$
24:     end for
25: end for
```

especially during the early learning stage (where the improvements need to take place), taking the predicted actions has two important disadvantages. First, the corrections and their estimates are coarse improvements that primarily aim to explore. The eventual performance is limited and at some point the actor will perform better and the predictor's influence needs to be scheduled away.

A second problem is that the predictor generalises from few feedback samples and its policy may not be very expressive. As a corollary, the variance in the interactions is reduced and this impedes the learning from this data. As clearly demonstrated in [28], learning from data generated by a stationary policy will cause for instability, presumably because of overfitting. In addition, we suspect that the `Adam` optimiser [29] may become over-confident in its gradient estimate (which is now artificially consistent) and raises at least some of the adaptive learning rates to an unstable value. In [28] the problems are overcome by collecting data with a random or pristine policy. Accordingly, we disable the predictor during the first N_p non-corrected samples. As a second countermeasure, we inject noise to the estimates with

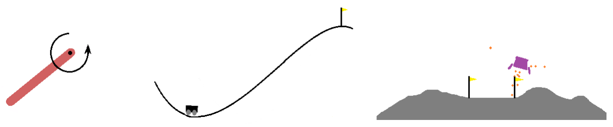

Fig. 3. From left to right: Pendulum-v0, MountaincarContinuous-v0 and LunarLanderContinuous-v2 from the OpenAI gym [31]. The respective goals in these underactuated problems is to swingup and balance, drive up the mountain and gently land between the flags.

variance $\sigma_{\hat{a}_c}$ (line 6), such that the original distribution is somewhat restored and the variance problems partly alleviated. Finally, note that in successful actor-critic learning, the critic learns faster than the actor [30]. We can therefore interleave $\hat{a}_c$ and a_p using a Q-filter that selects the action with the greatest value (line 7). Besides the retaining of buffer variance, emphasis will now be scheduled towards the actor upon its convergence so the Q-filter also solves the first problem (transcending the predictor performance). Because the critic needs to be learned before it can correctly schedule, it is enabled after N_Q samples. Note that, in contrast to the use of direct scheduling heuristics [19], there is a wide range in which N_Q and N_P are successfully set (Fig. 6).

IV. Implementation and Evaluation

Our code is available at github.com/janscholten/ppmp. All five neural networks (2 for the actor, 2 for the critic, and one for the predictor) are of size (400, 300) and use ReLU activation (except for hyperbolic tangent output layers that delimit actions within their bounds). We train with `Adam` [29], using learning rates of 0.002 for the critic 0.005, 0.0001 for the actor and 0.0002 for the predictor. The actor has $K = 10$ heads. The soft target mixing factor $\tau = 0.003$. The initial variance in the network initialisations is 0.001. The buffers B and B_c have size 1M and 1600 respectively and the minibatch size is 64. The discount rate is $\gamma = 0.99$. The OU-process has volatility 0.3, damping 0.15 and timestep 0.01. The selector and predictor have (as a fraction of the action range per channel) resolution $d = 0.125$, scale $c_s = 0.5$ and variance $\sigma_{\hat{a}_c} = 0.025$. The correction variance is set to $\Sigma_{hh} = 1 \cdot 10^{-8}$ The predictor and Q-filter are enabled after $N_P = 1500$ and $N_Q = 4000$ samples respectively.

For benchmarking purposes we regard the problem set in Fig. 3. The continuous state space of the pendulum environment consists of x- and y-positions of the tip and the angular velocity. The control input is the applied torque. Negative reward is given both for displacement from the upright equilibrium and for applied torque. An episode lasts 200 timesteps. The mountain car's state space consists of position and speed and it is controlled by a (reverse) driving force (again, all continuous). When the car reaches the flag, the episode is over and a reward of 100 is obtained, discounted by the cumulative squared action. The state space of the lunar lander has eight dimensions, both continuous (positions/velocities) and binary (leg contact). It is controlled with two inputs, one for the main engine and a second for the steering rockets. An episode ends upon soft landing to rest (100 points) or crashing (-100), and navigation yields

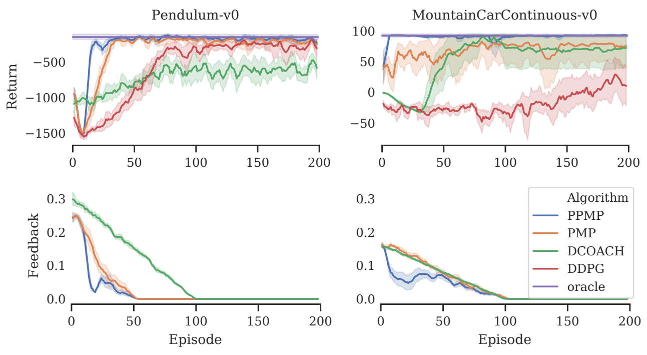

Fig. 4. Our methods PPMP and its ablation PMP (without prediction) outperform all baselines. Depicted is the moving average of ten evaluations (window size 5) along with the feedback rate.

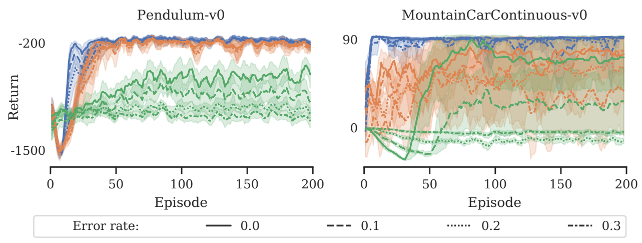

Fig. 5. In case of erroneous feedback, our method proves to be robust and the sample efficiency is hardly affected. The curves with perfect feedback are equal to those in Fig. 4 and the same legend applies.

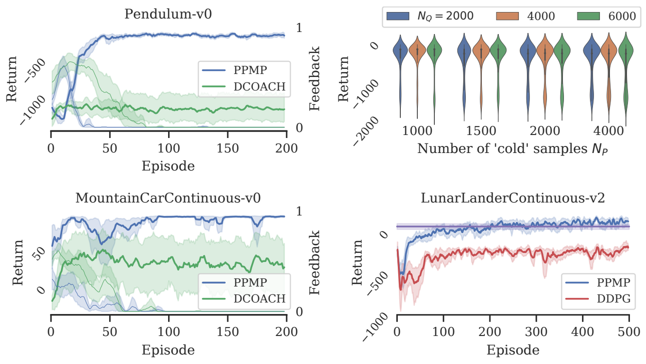

Fig. 6. **Left:** Performance with human participants, averaged over 12 experiments (feedback rate is the thinner line). **Top right:** A sensitivity analysis of the introduced hyperparameters. **Bottom right:** PPMP learns to land, and thereby outperforms the oracle that only knows how to fly.

between 100 and 140 points. The applied actions are discounted. Unsolved episodes terminate after 1000 timesteps.

To account for inconsistencies in human feedback [19], we use synthesised feedback (oracle). To study applicability, we additionally test with human participants. The oracle compares a with a converged policy. We apply the assumed distance d as a threshold for the feedback, but not for the `DCOACH` implementation (the performance would be unfairly hindered by an assumption that the authors do not make in their work). The feedback rate is controlled with a biased coin-flip and annealed over time. To infer robustness we apply erroneous feedback, implemented as in [12].

The results with human participants were obtained from three participants in the age of 20 to 30 with different backgrounds. For each algorithm, they were given a single demonstration and a test run possibility to get familiar with the interface. The subsequent four runs were recorded.

We evaluate PPMP and its ablation PMP (without the predictor) and compare with DDPG [24] and DCOACH [32] (a non-RL deep method that learns from corrective feedback only). Implementations are from P. Emami and R Pérez Dattari on Github.com. We generated ten random seeds (with another random generator) which we applied to ten runs of each algorithm respectively, such that the environments feature equal stochasticity for the different implementations. We evaluate the results on four criteria: sample efficiency, feedback efficiency, final performance, and robustness to erroneous feedback.

V. Results

Fig. 4 shows that our methods outperform other algorithms in every respect, and whereas the baselines seem to suit one problem in particular, PPMP is consistent. DDPG agents only learn good policies within the range of 200 episodes in the Pendulum problem, for the other environments its poor sample efficiency is even more outspoken. From a comparison with PMP, we infer that the predictor module provides for better sample efficiency, more consistent performance, greater final performance and a reduction of the feedback demand. Fig. 5 shows the effect of erroneous feedback. In virtue of value-based learning, our methods prove very robust to erroneous feedback and there is no serious impediment of final performance. In contrast, DCOACH is greatly affected and fails for error rates beyond 10%.

In Fig. 6 additional results are presented. On the left, results from human participants confirm our findings from simulated feedback. DCOACH obtains a considerable amount of feedback, but inconsistent feedback causes failure nonetheless. PPMP is more feedback efficient, learns fast, and consistently attains great final performance. There are some set-backs in performance in the Mountain Car problem, presumably a result of participants that assume the learning is finished after some early success.

Top right in Fig. 6 is a sensitivity analysis for the new hyperparameters N_Q and N_p. We compare the distribution of the return during the first 15000 timesteps in the Pendulum domain. As stated in Sec. III-D the new parameters neither require meticulous tuning nor cause brittleness.

Next, in the bottom right, let us consider a typical use case where the feedback has limited performance and is not able to fully solve the problem itself. This scenario is different from the previous studies (with erroneous feedback), where incidental mistakes may have been overcome by low-pass dynamics and generalisation but corrections eventually extended to the end-goal. We use the Lunar Lander environment, where the oracle is now partial as it knows how to fly but can not land. The sequel of the problem is thus left to the agent. It is emphasised that the reward function of this environment stresses stable operation by assigning great negative reward to crashes. Only the last 100 reward units that our method obtains correspond to having learned to land properly. As such, our method allows to solve a problem that is otherwise not feasible.

VI. Conclusion

This work discusses how binary corrective feedback may be used as a probabilistic exploration signal in DRL in order to improve its sample efficiency. By slight modification of an off-policy algorithm (here DDPG) the uncertainty in the policy was obtained and coupled with the magnitude of the correction induced by the feedback. To generalise corrections and improve memorisation, a predictor network provides estimates of the corrected policy, which can substitute for the actual policy when it increases the value of the state-action pair (especially during early learning). Our method, Predictive Probabilistic Merging of Policies (PPMP), is easily implemented and makes realistic assumptions about the feedback: it does not need to be a full demonstration, expertise may be limited, we do not assume feedback is abundant, neither do we require simulation or post-processing. Nevertheless, PPMP consistently improves on sample efficiency, final performance and robustness in comparison to pure RL (DDPG) and learning from corrections only (DCOACH), both for simulated and human feedback.

A first topic further research should address, is how PPMP carries over to real-world scenarios. Although the scalability and robustness are promising, the applicability is not yet proven by this work. From the possible extensions to this study, an exciting avenue would be to refine the probabilistic part. In particular, we would like to dispose of the Gaussian approximation and connect with full distributional learning [33], [34], possibly combined with uncertainty-handling estimation of human feedback [35]. This could give better estimates of the abilities and allow for more sophisticated action selection, e.g., posterior or Thompson sampling [26].

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, second ed., 2018.

[2] R. Pinsler, R. Akrour, T. Osa, J. Peters, and G. Neumann, "Sample and feedback efficient hierarchical reinforcement learning from human preferences," in *IEEE Int. Conf. Robotics & Automation (ICRA)*, 2018.

[3] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *Int. Conf. Intelligent Transp. Syst. (ITSC)*, 2018.

[4] J. Camhi, "AI in supply chain and logistics," *Business Insider Intelligence*, 2018.

[5] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," 2017. arXiv:1709.06560 [cs.LG].

[6] D. Ernst, G.-B. Stan, J. Goncalves, and L. Wehenkel, "Clinical data based optimal STI strategies for HIV: A reinforcement learning approach," in *IEEE Conf. Decision and Control (CDC)*, 2007.

[7] Y. Zhao, M. R. Kosorok, and D. Zeng, "Reinforcement learning design for cancer clinical trials," *Statistics in Medicine*, vol. 28, no. 26, pp. 3294–3315, 2009.

[8] American Diabetes Association, "Economic costs of diabetes in the U.S. in 2017," *Diabetes Care*, 2018.

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

[10] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[11] W. B. Knox and P. Stone, "Reinforcement learning from simultaneous human and MDP reward," in *Int. Conf. Autonomous Agents and Multiagent Syst. (AAMAS)*, 2012.

[12] C. Celemin, J. Ruiz-del Solar, and J. Kober, "A fast hybrid reinforcement learning framework with human corrective feedback," *Autonomous Robots*, vol. First Online, 2018.

[13] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," 2017. arXiv:1709.10089 [cs.LG].

[14] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothrl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017. arXiv:1707.08817 [cs.AI].

[15] M. Monfort, M. Johnson, A. Oliva, and K. Hofmann, "Asynchronous data aggregation for training end to end visual control networks," in *Conf. Autonomous Agents and MultiAgent Syst. (AAMAS)*, 2017.

[16] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances Neural Information Processing Syst. (NIPS)*, 2017.

[17] H. B. Suay and S. Chernova, "Effect of human guidance and state space size on interactive reinforcement learning," in *Int. Symp. Robot and Human Interactive Communication (RO-MAN)*, 2011.

[18] W. B. Knox and P. Stone, "Combining manual feedback with subsequent mdp reward signals for reinforcement learning," in *Int. Conf. Autonomous Agents and Multiagent Syst. (AAMAS)*, 2010.

[19] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Advances Neural Information Processing Syst. (NIPS)*, 2013.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Int. Conf. Machine Learning (ICML)*, 2018.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

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

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *Int. Conf. Learning Representations (ICLR)*, 2016.

[25] Y. Gal, *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[26] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped *DQN*," in *Advances Neural Information Processing Syst. (NIPS)*, 2016.

[27] C. Rupprecht, I. Laina, R. DiPietro, and M. Baust, "Learning in an uncertain world: Representing ambiguity through multiple hypotheses," in *IEEE Int. Conf. Computer Vision (ICCV)*, 2017.

[28] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "The importance of experience replay database composition in deep reinforcement learning," in *Deep Reinforcement Learning Workshop, NIPS*, 2015.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. arXiv:1412.6980 [cs.LG].

[30] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances Neural Information Processing Syst. (NIPS)*, 2000.

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

[32] R. Pérez Dattari, C. Celemin, J. Ruiz Del Solar, and J. Kober, "Interactive learning with corrective feedback for policies based on deep neural networks," in *Int. Symp. Experim. Robotics (ISER)*, 2018.

[33] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Int. Conf. Machine Learning (ICML)*, 2017.

[34] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap, "Distributional policy gradients," in *Int. Conf. Learning Representations (ICLR)*, 2018.

[35] D. Wout, J. Scholten, C. Celemin, and J. Kober, "Learning Gaussian policies from corrective human feedback," 2019. arXiv:1903.05216 [cs.LG].

Bibliography

Amershi, S., Cakmak, M., Knox, W. B., and Kulesza, T. (2014). Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. (2018). Distributional policy gradients. In *International Conference on Learning Representations*.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Camhi, J. (2018). AI in supply chain an logistics. *Business Insider Intelligence*.

Cederborg, T., Grover, I., Isbell, C. L., and Thomaz, A. L. (2015). Policy shaping with human teachers. In *IJCAI*, pages 3366–3372.

Celemin, C., Maeda, G., del Solar, J. R., Peters, J., and Kober, J. (2018). Reinforcement learning of motor skills using policy search and human corrective advice. *The International Journal of Robotics Research*, unpublished.

Celemin, C., Maeda, G., Kober, J., and Ruiz-del Solar, J. (2017). Human corrective advice in the policy search loop. *Not published*.

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

Celemin, C. and Ruiz-del Solar, J. (2015b). Interactive learning of continuous actions from corrective advice communicated by humans. In *Robot Soccer World Cup*, pages 16–27. Springer.

Celemin, C. and Ruiz-del Solar, J. (2018). An interactive framework for learning continuous actions policies based on corrective feedback. *Journal of Intelligent & Robotic Systems*, pages 1–21.

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

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4302–4310.

de Bruin, T., Kober, J., Tuyls, K., and Babuška, R. (2015). The importance of experience replay database composition in deep reinforcement learning. In *Deep Reinforcement Learning Workshop, NIPS*.

de Bruin, T., Kober, J., Tuyls, K., and Babuška, R. (2018). Experience selection in deep reinforcement learning for control. *Journal of Machine Learning Research*, 19(9):1–56.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338.

Ernst, D., Stan, G.-B., Goncalves, J., and Wehenkel, L. (2007). Clinical data based optimal sti strategies for hiv: A reinforcement learning approach. *Proceedings of the IEEE Conference on Decision and Control*, pages 667 – 672.

Ferdowsi, A., Challita, U., Saad, W., and Mandayam, N. B. (2018). Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 307–312. IEEE.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15:315–323. cited By 1002.

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.

Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1856–1865.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2017). Deep reinforcement learning that matters. *CoRR*, abs/1709.06560.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

IFR (2017). Executive summary of the world robotics 2017 congress on industrial robots. *International Federation of Robotics*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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.

Knox, W. B. and Stone, P. (2010). Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems.

Knox, W. B. and Stone, P. (2012). Reinforcement learning from simultaneous human and mdp reward. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 475–482. International Foundation for Autonomous Agents and Multiagent Systems.

Knox, W. B., Stone, P., and Breazeal, C. (2013). Training a robot via human feedback: A case study. In *International Conference on Social Robotics*, pages 460–470. Springer.

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

Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

León, A., Morales, E. F., Altamirano, L., and Ruiz, J. R. (2011). Teaching a robot to perform task through imitation and on-line feedback. In *Iberoamerican Congress on Pattern Recognition*, pages 549–556. Springer.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. *International Conference on Learning Representations.*

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321.

Lin, Z., Harrison, B., Keech, A., and Riedl, M. O. (2017). Explore, exploit or listen: Combining human feedback and policy model to speed up deep reinforcement learning in 3d worlds. *arXiv preprint arXiv:1709.03969.*

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

MacGlashan, J., Ho, M. K., Loftin, R., Peng, B., Roberts, D., Taylor, M. E., and Littman, M. L. (2017). Interactive learning from policy-dependent human feedback. *arXiv preprint arXiv:1701.06049.*

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., and Ostrovski, G. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2017). Overcoming exploration in reinforcement learning with demonstrations. *CoRR*, abs/1709.10089.

Ngo, H., Luciw, M., Nagi, J., Forster, A., Schmidhuber, J., and Vien, N. A. (2014). Efficient interactive multiclass learning from binary feedback. *ACM Trans. Interact. Intell. Syst.*, 4(3):12:1–12:25.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped *DQN*. In *Advances in neural information processing systems*, pages 4026–4034.

Peng, B., Loftin, R., MacGlashan, J., Littman, M. L., Taylor, M. E., and Roberts, D. L. (2015). Language and policy learning from human-delivered feedback. In *Proceedings of the Machine Learning for Social Robotics workshop (ICRA).*

Peng, B., MacGlashan, J., Loftin, R., Littman, M. L., Roberts, D. L., and Taylor, M. E. (2016). A need for speed: Adapting agent action speed to improve task learning from non-expert humans. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 957–965. International Foundation for Autonomous Agents and Multiagent Systems.

Pérez Dattari, R., Celemin, C., Ruiz Del Solar, J., and Kober, J. (2018). Interactive learning with corrective feedback for policies based on deep neural networks. In *International Symposium on Experimental Robotics (ISER)*.

Pinsler, R., Akrour, R., Osa, T., Peters, J., and Neumann, G. (2018). Sample and feedback efficient hierarchical reinforcement learning from human preferences. In *IEEE Int. Conf. Robotics Automation (ICRA)*.

Rastogi, D., Koryakovskiy, I., and Kober, J. (2018). Sample-efficient reinforcement learning via difference models. In *Third Machine Learning in Planning and Control of Robot Motion Workshop at IEEE International Conference on Robotics and Automation (ICRA)*.

Rupprecht, C., Laina, I., DiPietro, R., and Baust, M. (2017). Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3611–3620. IEEE.

Scholten, J., Wout, D., Celemin, C., and Kober, J. (2019). Deep reinforcement learning with feedback-based exploration. *ArXiv*. arXiv:1903.06151 [cs.LG], [cs.AI], [stat.ML].

Senft, E., Lemaignan, S., Baxter, P. E., and Belpaeme, T. (2017). Leveraging human inputs in interactive machine learning for human robot interaction. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 281–282. ACM.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

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

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

Sutton, R. S., Barto, A. G., and Williams, R. J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems*, 12(2):19–22.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Thomaz, A. L. and Breazeal, C. (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.

Thomaz, A. L. and Breazeal, C. (2008). Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Tsitsiklis, J. N. and Van Roy, B. (1997). Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pages 1075–1081.

Vecerík, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. A. (2017). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR, abs/1707.08817*.

Verhaegen, M. and Verdult, V. (2007). *Filtering and system identification: a least squares approach*. Cambridge university press.

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

Warnell, G., Waytowich, N., Lawhern, V., and Stone, P. (2017). Deep tamer: Interactive agent shaping in high-dimensional state spaces. *arXiv preprint arXiv:1709.10163*.

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

Wiering, M. and Van Otterlo, M. (2012). Reinforcement learning. *Adaptation, learning, and optimization*, 12.

Wirth, C. (2017). *Efficient Preference-based Reinforcement Learning*. PhD thesis, Technische Universität.

Wout, D., Scholten, J., Celemin, C., and Kober, J. (2019). Learning gaussian policies from corrective human feedback. *ArXiv*. arXiv:1903.05216 [cs.LG].

Zhao, Y., Kosorok, M. R., and Zeng, D. (2009). Reinforcement learning design for cancer clinical trials. *Statistics in medicine*, 28(26):3294–3315.

Glossary

DL	Deep Learning	3
IRL	Interactive Reinforcement Learning	3
MDP	Markov Decision Process	3
RL	Reinforcement Learning	iii
COACH	COrrective Advice Communicated by Humans	8
DCOACH	Deep COrrective Advice Communicated by Humans	25
DDPG	Deep Deterministic Policy Gradient	iii
TAMER	Training an Agent Manually via Evaluative Reinforcement	8
PMP	Probabilistic Merging of Policies	51
PPMP	Predictive Probabilistic Merging of Policies	51