
Learning from Human Feedback: A Comparison of Interactive Reinforcement Learning Algorithms

Von Menschlichem Feedback Lernen:

Ein Vergleich von Interactive Reinforcement Learning Algorithmen

Master-Thesis von Maximilian Kircher aus Hadamar

Dezember 2019



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Learning from Human Feedback:

A Comparison of Interactive Reinforcement Learning Algorithms

Von Menschlichem Feedback Lernen:

Ein Vergleich von Interactive Reinforcement Learning Algorithmen

Vorgelegte Master-Thesis von Maximilian Kircher aus Hadamar

1. Gutachten: Prof. Dr. Jan Peters

2. Gutachten: Dr. Carlo D'Eramo

3. Gutachten: M.Sc. Dorothea Koert

Tag der Einreichung:

Erklärung zur Master-Thesis

Erklärung zur Abschlussarbeit gemäß § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Maximilian Kircher, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

In the submitted thesis the written copies and the electronic version are identical in content.

Datum / Date:

Unterschrift / Signature:

Abstract

Reinforcement Learning potentially provides a powerful tool for self improvement of future robots. However, in contrast to learning in simulation, on real robots it is much more important to be sample efficient. When applying Reinforcement Learning, especially the recently successful Deep Reinforcement Learning, usually many interactions with their environment are required to learn. This is not practicable when training with a real system. Another challenge is the design of good reward functions to enable correct and fast learning of the task at hand. In particular, when robots shall interact with us in our everyday life, it would regularly be necessary to learn new tasks or to adapt learned behaviors to new circumstances. Directly integrating the human in the training could hereby help to become more sample efficient in training, and to adapt the reward to human preferences. All in all, human interaction could thereby help to bring Reinforcement Learning to real robot applications.

This thesis presents a general framework that can incorporate different kinds of human interaction and the environmental reward in an Interactive Reinforcement Learning algorithm. Through this, it simplifies the combination of several, interactive and not interactive, algorithms. Various such combinations of algorithms are tested in experiments with different learning environments and both, simulated human feedback and feedback from real users. The experimental evaluations showed that human interaction could nearly always speed up the learning process and even enables the learning of tasks that require very extensive exploration, and thus could not be learned with the Reinforcement Learning baseline. In most experiments with real humans, the use of advice worked better than giving feedback.

Zusammenfassung

Reinforcement Learning bietet potentiell ein mächtiges Werkzeug für die eigenständige Verbesserung zukünftiger Roboter. Allerdings ist es für das Training echter Roboter, im Gegensatz zu Training in Simulationen, deutlich wichtiger Daten effizient zu nutzen. Bei der Anwendung von Reinforcement Learning, insbesondere des zuletzt erfolgreichen Deep Reinforcement Learning, werden meist viele Interaktionen mit der Umgebung zum Lernen benötigt. Dies ist jedoch für das Trainieren auf einem realen System nicht praktikabel. Eine weitere Herausforderung ist das Design guter Rewardfunktionen, um korrektes und schnelles Lernen für die dazugehörige Aufgabe zu ermöglichen. Insbesondere wenn Roboter in unserem Alltag mit uns interagieren sollen, wäre es regelmäßig notwendig neue Aufgaben zu erlernen oder bereits Gelerntes an neue Umstände anzupassen. Den Menschen direkt in das Training zu integrieren könnte dabei helfen mit wenigen Interaktionen lernen zu können und den Reward an die Präferenzen des Nutzers anzupassen. Insgesamt könnte menschliche Interaktion dadurch dabei helfen Reinforcement Learning auf echten Robotern anzuwenden.

Diese Abschlussarbeit präsentiert ein allgemeines Framework, das verschiedene Arten menschlicher Interaktion und den Reward der Umgebung in einen Interactive Reinforcement Learning Algorithmus einbindet. Dadurch vereinfacht es die Kombination mehrerer, interaktiver und nicht interaktiver, Algorithmen. Verschiedene solche Kombinationen von Algorithmen werden in Experimenten mit verschiedenen zu erlernenden Umgebungen und, sowohl simuliertem Feedback, als auch Feedback von echten Benutzern, getestet. Die Evaluierung der Experimente hat gezeigt, dass menschliche Interaktion den Lernprozess fast immer beschleunigen konnte und sogar das Trainieren für Aufgaben die eine sehr umfassende Exploration der Umgebung erfordern und daher nicht durch den Reinforcement Learning Vergleichsalgorithmus erlernt werden konnten, ermöglichte. In den meisten Experimenten mit echten Nutzern funktionierte das Vorschlagen von Aktionen besser als deren Bewertung.

Acknowledgments

First, I want to thank my supervisors Dorothea Koert and Carlo D'Eramo for their support from the first idea to the completion of this thesis. Their advice and feedback helped me much.

Besides, I want to thank my family that supported me during my whole time at university.

And last: Thanks to everyone who helped me performing experiments with real users by participating in the user studies.

Contents

1. Introduction	1
1.1. Research Question	2
1.2. Thesis Structure	2
2. Related Work and Foundations	4
2.1. Related Work	4
2.1.1. Human Reward	4
2.1.2. Guidance	5
2.1.3. Other Kinds of Information	5
2.2. Foundations	6
2.2.1. Reinforcement Learning Baseline	6
2.2.2. TAMER	6
DeepTAMER	6
TAMER with Environmental Reward	6
2.2.3. Supervised Reinforcement Learning	7
3. Combination of Interactive Reinforcement Learning Algorithms	8
3.1. Fusion of Learned Models	9
3.1.1. TAMER	9
3.1.2. Advice	9
3.1.3. Supervised Reinforcement Learning (SRL)	10
3.1.4. Simultaneous TAMER (Sim TAMER)	10
3.1.5. Simultaneous Advice (Sim Advice)	10
3.1.6. Simultaneous (Sim)	10
3.1.7. Advised TAMER (Sup Advice)	11
3.1.8. Supervised Advice	11
3.2. Model Updates	11
3.2.1. $Q(f)$ - Reinforcement Learning Update	11
3.2.2. $F(f)$ - TAMER Update	12
3.2.3. $A(f)$	13
4. Implementation Details	14
4.1. Learning Algorithms	14
4.2. User Interface	16
5. Experimental Evaluation	18
5.1. Learning Environments	18
5.1.1. Gridworld	18
5.1.2. Taxi like Gridworld with Holes	19
5.1.3. Mountain Car	20
5.1.4. Lunar Lander	20
5.2. Simulation Results	21
5.2.1. Gridworld	21
5.2.2. Taxi like Gridworld with Holes	22
5.2.3. Mountain Car	23
5.3. User Studies	24
5.3.1. Mountain Car (1)	24
5.3.2. Mountain Car (2)	26
5.3.3. Lunar Lander - Expert Trainer	29
5.3.4. Lunar Lander - Non-Expert Trainer	32
5.3.5. Taxi like Gridworld with Holes	32

6. Conclusion and Future Work	35
6.1. Research Questions	35
6.2. Future Work	36
Bibliography	37
A. Transform State Features with Actions to State Action Features	40
B. User Study Forms	41
B.1. English User Study Forms	41
B.1.1. Mountain Car (1)	42
B.1.2. Mountain Car (2)	49
B.1.3. Lunar Lander	53
B.1.4. Taxi Like Gridworld	57
B.2. German User Study Forms	62
B.2.1. Mountain Car (1)	63
B.2.2. Mountain Car (2)	70
B.2.3. Lunar Lander	74
B.2.4. Taxi Like Gridworld	78

Figures and Tables

List of Figures

1.1. Diagram of Interactive Reinforcement Learning	1
2.1. Interactive Reinforcement Learning Framework of [1]	5
3.1. Overview of the Proposed Framework	8
4.1. Structure of the Code	14
4.2. User Interface	16
5.1. The Gridworld Environment	18
5.2. The Taxi Environment	19
5.3. The Mountain Car Environment	20
5.4. The Lunar Lander Environment	20
5.5. Simple Gridworld Experiments	21
5.6. Taxi like Gridworld with Holes Experiments	22
5.7. Mountain Car Experiments without Environmental Reward	23
5.8. Mountain Car Experiments with Environmental Reward	24
5.9. Results of the First User Study	25
5.10. Results of the Second User Study	28
5.11. Comparison of Interaction Strategies	29
5.12. Results of expert training on Lunar Lander without environmental reward	30
5.13. Results of expert training on Lunar Lander with environmental reward	31
5.14. Number of Interactions on the Taxi Like Gridworld	33
5.15. Evaluation of Learned Models for the Taxi like Gridworld Environment	34

List of Tables

2.1. Overview of Literature that Propose Different Ways to Give Human Feedback	4
5.1. Settings for the Experiments on the Gridworld Environment	22
5.2. Learning Rates for Experiments on the Mountain Car Environment	23
5.3. Results of Expert Training on Lunar Lander	31
5.4. Results of Non-Expert Training on Lunar Lander	32

1 Introduction

There is the vision, that robots shall be able to help us in our everyday lives. But to be able to do that, they will have to adapt to different users and different working environments. One example is the caring for elderly people: Some of the patients may need help with specific tasks that can differ much over time and between situations. Besides, the robots operate in their homes. A robot would have to adapt to this specific environment, as every house is different. Because such details can probably not be implemented beforehand and even change over time, the robot has to be able to adapt its behavior during operation.

Reinforcement Learning could be a way to attempt this problem. There has been big progress in the area of Reinforcement Learning, especially in Deep Reinforcement Learning (see e.g. [2, 3, 4] and many more), in recent years. The found algorithms are regularly applied to simulated robotic tasks as the mujoco ([5]) tasks in the OpenAI gym¹. In those standard simulated environments, there are predefined reward functions available. However, when a Reinforcement Learning technique shall be applied to a new task, a new reward function has to be defined. Defining good reward functions is often not easy, especially when dealing with real robots and when the users, who are usually no experts, shall be able to train the robot. Furthermore, Deep Reinforcement Learning algorithms typically need very many interactions with their environment. This is not feasible when robots shall learn in the real world. Thus, the sample complexity and learning times would need to be reduced.

There are different approaches to enable the usage of Reinforcement Learning without a predefined reward function. First, there is the field of Inverse Reinforcement Learning ([6]), where a reward function is learned from expert demonstrations (e.g. [7, 8]). This is part of the Learning From Demonstrations research area ([9]). One problem is, that it may be hard for humans to provide good demonstrations of a task. Again this is a problem, especially when dealing with real robots (see [10]), and when demonstrations shall be given by users that are no experts.

Another way to train an agent without a predefined reward function is to use human feedback instead, also referred to as Interactive Reinforcement Learning ([11]). In this approach, it is not clear in which way the human feedback can be used best because there are different ways to give feedback and also different ways to interpret the given feedback. In general, human feedback is quite different from a classical reward function and this has to be considered when it shall be used to train an agent. The Interactive Reinforcement Learning idea also includes giving advice to the agent to guide him through the learning. A diagram that shall give an intuition about the Interactive Reinforcement Learning in contrast to the classical Reinforcement Learning can be found in Figure 1.1.

Intuitively it makes sense that human interaction might improve the training of autonomous agents due to the human's prior knowledge about the world. On the other hand, the interaction with a human is costly and thus should be minimized. This is still valid when no reward function is available and only information from the human shall be used for the training. For problems with a high frequency of actions, it is not possible for humans to judge each state action pair. Besides, if many rollouts are needed for the training, it might be also unfeasible to give feedback for each state action pair. Therefore, the human reward is typically provided sparsely and has to be distributed over multiple time steps, if it is given per state action pair. An alternative is to only judge whole episodes or parts of

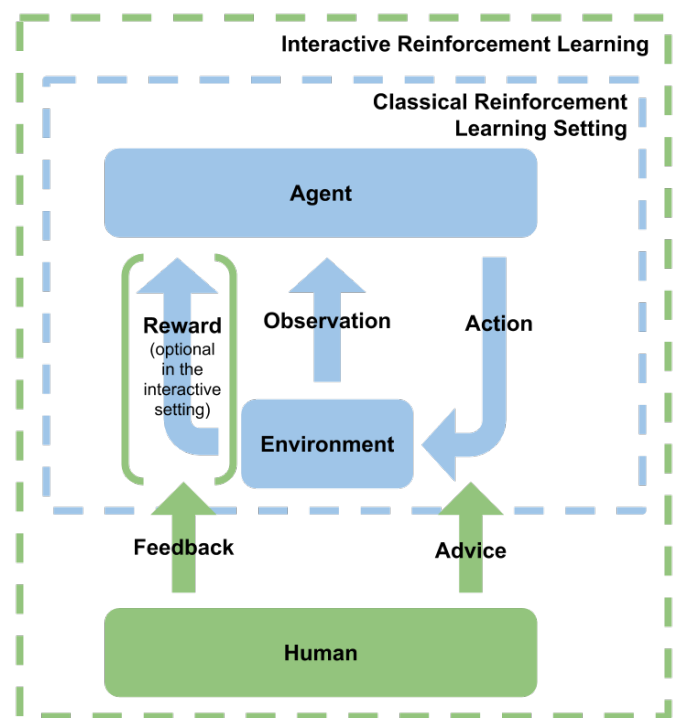


Figure 1.1.: Diagram of Interactive Reinforcement Learning
The upper, blue part depicts the classical Reinforcement Learning setting: The agent performs actions in the environment and receives an encoding of the current state and a reward in return. This is extended by the human interaction, shown in green, to get a diagram of the interactive reinforcement learning idea.

¹ <https://github.com/openai/gym>

episodes. Another approach to reducing human interaction is to compute when the agent most probably really needs feedback and query it from the user only then.

There are already several Interactive Reinforcement Learning algorithms, but they differ inter alia in the way the feedback is given, and it is not clear which way is best. For example, a user can say whether something was good, how good it was, or whether it was better than something else. You can find an overview of different ways to give feedback in Table 2.1.

Next to the way to give feedback, there are also multiple ways to interpret human feedback. For example, it can be assumed to be similar to a classical reward function. On the other hand, one can argue that humans already incorporate the long term cost of an action to their reward and thus the human reward would be more similar to a state action value function. Because human feedback is often relative to the agent's behavior that was seen before, it is also argued, that the human reward can be seen as Advantage function. Especially when using binary rewards, it is frequently assumed that human feedback tells whether an action was correct or not.

Besides giving a general numeric feedback, a human could for many tasks also tell the agent, what alternative might have been better than the performed action. Another way to incorporate human advice is to tell the agent what would be a good action to perform next. Here a good action could be an action that is important for exploration or that is really a good action that should be performed in the long term. This advising actions can be related to the Learning From Demonstrations area. Thus, there is also the problem that it might be hard to give good advice. However, there is the difference that an advice is typically not a whole demonstration but just a single action that suits a given situation.

It can be seen that there are plenty of ways to interpret the human information and it is not trivial to see, which one is best. Different interpretations and techniques might even be important for different problems and environments. All in all, learning from human reward is an important approach to bring robots and Reinforcement Learning to real world applications with human robot interaction. On the other hand, there are important differences to classical Reinforcement Learning and many ways to attempt them. To make further progress in this area, it is important to find the advantages and disadvantages of different approaches.

1.1 Research Question

The described problem setting leads to different research questions, that shall be tackled in this thesis.

1. Can human feedback enable or improve the training of agents?

There has been already work that demonstrated that human feedback during the learning of an agent was helpful. Besides, there is also work where agents learn only from human feedback, how to solve a task. So the first goal of this thesis is to reproduce such results and extend them to other problems.

2. How can different ways of interaction and different interpretations of human feedback be beneficial for the training?

In the existing literature of Interactive Reinforcement Learning, there were proposed different ways for human trainers to provide feedback (see Table 2.1). Besides, the feedback can be interpreted to represent different kinds of information. Another way of human interaction that has to be considered is to give advice instead of feedback. It is not clear which of those possibilities should be preferred.

3. Is there a clear relationship between different problem settings and corresponding best working methods?

It is not only unclear, which way of human interaction and its interpretations are best, but it is possible, and likely, that there is not a single perfect way that fits all problems. Thus it would be interesting, to find a relation between the problem setting and the best working approach.

1.2 Thesis Structure

The rest of the thesis consists of the following chapters:

Chapter 2: An overview of existing related literature is given. This includes human reward, guidance and other kinds of human interaction. Besides, the algorithms that were implemented and used for this thesis are introduced.

Chapter 3: A general framework for the combination of different kinds of human interaction and environmental reward is introduced. Furthermore, it is explained how different algorithms can be used in this Framework.

Chapter 4: Basic information about the structure of the implemented code is given. Additionally, some short code snippets are given as an illustration.

Chapter 5: Some examples of algorithms in the proposed framework were used to learn several tasks. To do that oracles that simulate human interaction and user studies with real humans were used. In this chapter, first, those experiments are explained and afterwards the results are evaluated.

Chapter 6: The thesis is concluded with a summary of the findings and an outlook on future work that could continue the work of this thesis.

2 Related Work and Foundations

In this chapter, an overview of the existing literature of Interactive Reinforcement Learning is given. Besides, the algorithms that were implemented and used in this thesis are introduced.

2.1 Related Work

There has been done much work that considers the learning of autonomous agents that involves a user in some way. This section gives an overview of the papers that are most related to this thesis. This includes feedback, advice and other kinds of information from human trainers.

2.1.1 Human Reward

A very natural way to avoid the need for a reward function, when there is a human that can interact with the robot, is to let the human directly give the reward for the behavior of the robot. In contrast to classical Reinforcement Learning, where typically a numerical reward for each state action pair is assigned, there are multiple possible kinds of feedback when interacting with a human. Some argue, that binary feedback (is an action or a trajectory OK or not) is easier to give for a human than some numeric feedback. On the other hand, numerical reward can yield more information. An alternative way is to compare different behaviors that were performed by the agent beforehand. Depending on the concrete task, it could be easier or harder to provide a comparison than to give numerical feedback. An overview of work that used different ways of human feedback can be seen in Table 2.1.

binary	numeric	comparison	kinesthetic	implicit
[12, 13, 14, 15, 16, 17]	[12, 18, 19, 20, 21]	[22, 21, 23, 24, 25]	[26, 22]	[27, 28, 29, 30]

Table 2.1.: Overview of Literature that Propose Different Ways to Give Human Feedback

There are multiple ways for humans to give feedback: It can be a binary (good / bad) decision. This kind is also captured in the more general case of giving a numerical value. There are also approaches where a value is computed out of data that is recorded about the human, who does only implicitly judge the agent's behavior in this case. When dealing with real physical robots, their behavior can be judged by providing an alternative for it. These approaches typically make use of the more general idea to let the human compare different behaviors and learn from this. Some examples of those different approaches are listed in this table.

Early work that considered learning from human reward was, for example [11, 31], where the human reward was used similar to the classical reward in Reinforcement Learning with a Q-learning algorithm. They find in user studies that humans often try to not only judge the past behavior of an agent but also to guide its future behavior. Thus, they also integrate explicit guidance to their learning algorithm. Besides, they find that humans typically change their strategy of giving rewards, during the training process. Because this is not common in classical Reinforcement Learning, they propose to consider it when learning from human reward. Although the approach they presented is an early and naïve one, it was already applied to a simple real robot task in [32].

Instead of using human reward the same as a classical reward function, in COACH ([18]) it is assumed to be the Advantage function and directly used to optimize the policy. This is inspired by actor critic algorithms as, for example, [4]. The feedback is distributed over multiple timesteps by using multiple eligibility traces ([33]).

Another way of incorporating human reward is to assume that it tells whether an action or episode was correct. Upon this, a probabilistic model of the belief over the correctness can be computed. In Advice ([14]) this belief is used to improve the belief, resulting from Bayesian Q-learning ([34]), on an additional classical reward signal. A different approach under this assumption is (I-)SABL ([15]), where the likelihood of policies is computed. A model of the human trainer is either learned (I-SABL) or assumed to be given (SABL). This model encodes whether the trainer tends to rather give feedback either when he is satisfied (tends to give positive feedback) or when he is not (tends to give negative feedback). Such a model can then be inverted to tell whether the trainer was probably satisfied or not when he did not give feedback for some action. Thus, in this method, the absence of human feedback also yields information. In [16] the algorithm of [15]

is used and the knowledge about the uncertainty over different actions is exploited by adapting the movement speed of the agent to it. This can accelerate the training time and make it more comfortable for the human trainer.

Another approach is to simply learn a model of the given feedback. This way, in contrast to modeling the correctness of actions, arbitrary kinds of numeric feedback can be handled. An early and successful example, that implements this idea, is the TAMER framework ([12]). It was also extended to deep neural networks in [19]. Both algorithms are explained in more detail in Section 2.2. In [20] a probabilistic model of the reward function is learned. The probabilistic model has again the advantage that the uncertainty of the model can be measured. This is exploited by querying the human to give feedback when the model uncertainty is high in some situation.

In preference based algorithms, there is not an absolute value assigned to the performed behavior, but multiple different behaviors are compared. A comprehensive survey of this area can be found in [35]. Examples for preference based algorithms are [22, 24], where reward models are learned from comparisons of trajectories. In preference based learning, it is sometimes also possible to show the robot by kinesthetic teaching how to improve the current attempt ([22]). Those demonstrations do not have to be optimal.

One recent approach tries to combine learning from positive examples and human feedback ([17]). In contrast to classical Inverse Reinforcement Learning no trajectories, but only images of states that shall be reached have to be provided. Those images are used to create a reward function. For those states where the algorithm is not certain about the reward, the human is queried for feedback.

Next to the approaches, where humans give explicit feedback, whether they liked some robot behavior or not, there is also work, that tries to extract the feedback directly from the human. This can be done either with EEG data ([28]) or by analyzing human behavior from video data ([29, 30]).

Because different methods to incorporate human reward have different advantages and drawbacks, [36] tries to combine different ones to get an optimal behavior. However, they assume that the human reward is given in addition to a classical Reinforcement Learning reward.

2.1.2 Guidance

The learning from human reward is related to human guidance during learning. An example of this is [37]. Here a given classical reward function is assumed, and the user can advise the agent which action to take. Then a positive reward signal is added to the environmental one for the state action pairs that were suggested by the user. As already mentioned in the previous section, [31] uses human guidance additionally to human reward.

Human guidance can also help the agent to explore the environment. For example in [38] human guidance is queried, which does not have to be optimal but just shall help the agent to explore the environment. Human guidance can also help to make the exploration during the learning of real robots safer. An example of this is [39], where the robot indicates the human, what he wants to do next. The human can prevent bad and potentially dangerous actions and suggest others. This human guidance is then used to improve the robot's policy. In [40] multiple channels of prior information can be combined to guide the agent. This algorithm will be explained in more detail in Section 2.2.3.

2.1.3 Other Kinds of Information

There are also other kinds of information, which can potentially help the agent to train, but which are used not that frequently. For example, one can start with a limited action space at the beginning and add actions that might be helpful during training as in [41]. Another example is [42], where the similarity between state action pairs can be defined and changed during the training. A more general framework for the human to interact during a Reinforcement Learning process is presented in [1]. The special thing about this approach is that it is completely independent of the used Reinforcement Learning algorithm. Some methods that can be used in this framework are that the feature encoding of states can be changed or actions are pruned during the learning process. Pruning actions, i.e. removing actions from the action space of the MDP, can be seen as the counterpart of adding actions during the training as proposed in [41]. The figure that the authors used in [1] to illustrate their view on Interactive Reinforcement Learning can be seen in Figure 2.1.

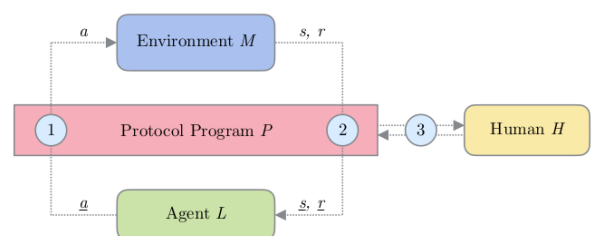


Figure 2.1.: Interactive Reinforcement Learning Framework of [1]

This figure is used in [1] to illustrate their view on Interactive Reinforcement Learning that includes many different types of human interaction.

2.2 Foundations

In this section, the algorithms that were implemented or directly influenced implementations in this thesis are introduced. How exactly they were used is explained in the next chapter (Chapter 3).

2.2.1 Reinforcement Learning Baseline

To compare the Interactive Reinforcement Learning approaches with classical Reinforcement Learning, baselines with a similar complexity are needed. Besides, some of the presented approaches include a Reinforcement Learning part. For both purposes, the following algorithms are used.

For a linear model complexity we use SARSA ([43]) with eligibility traces (SARSA(λ) [44]). With this algorithm, a state action value function can be learned. This is necessary for the combination of human interaction with the environmental reward in the approaches that will be presented later. In our implementation, the value function is represented by a linear model based on a feature representation of the state action pairs. The SARSA algorithm is a classical approach to this task. In the rest of this thesis, it will just be called SARSA for simplicity.

To learn a state action value function with a deep neural network, the DQN algorithm ([2]) is the classical method that can be applied. We use the Double DQN version of this algorithm ([45]), but for simplicity, it will be called DQN in the rest of this thesis.

2.2.2 TAMER

A classical algorithm, that is often used as a baseline for Interactive Reinforcement Learning, is TAMER ([12]). It can process numeric as well as binary human feedback, which is assumed to correspond to the value of state action pairs. It assumes some feature representation for the state action pairs. In the algorithm, a linear reward model over those features is learned. To do that the model is updated when the human gives some feedback. This way it can handle changes in the way the human gives feedback during the training. The learned model can then be used similar to a classical Q-table. Thus, actions are chosen to maximize the corresponding value of the reward model. TAMER uses the following loss to optimize the reward model: The features of all previous state action pairs are weighted by a guess how much they influenced the feedback. Those weighted features are summed up, and one value is computed for them. The difference between this computed value and the actual human feedback is the loss that is minimized. To compute how much single actions influenced the feedback, there is a probability distribution over the time before the feedback was given. For each action, the probability mass of the time that passed until the next action is used as a weight. For their results, they use a uniform distribution between some minimal and maximal reaction time.

DeepTAMER

There exists also a deep version of TAMER ([19]). To compute features, they propose to train an autoencoder neural network on the state space of the used environment. The encoder part of the network can then be used for the feature extraction. This is especially useful to compress very high dimensional state spaces, for example, images. In contrast to TAMER, they do not include the action for the feature computation but compute one value for each possible action in the reward model. For our experiments, we do not use the autoencoder features, because we do not use environments that encode their states as images. However, we do compute different values for different actions and do not include the actions in the feature computation. For the reward model, they also use a neural network. Another difference to TAMER is, that they save the feedback in a replay memory and use that for replay updates. This shall reduce the needed amount of human feedback because neural networks typically need many update steps to be optimized. On the other hand, this can become problematic, when the human trainer changes his way to give feedback during training. Deep TAMER also differs from TAMER in the used loss function. They explicitly compute the squared loss for each considered state action pair and its corresponding human feedback and weight them afterwards.

TAMER with Environmental Reward

The TAMER framework is good when there is no reward function available because it enables to learn a task only by human feedback. But in the scenario that some basic reward function exists and human feedback shall only improve the training, TAMER can not be used directly. Because of that, in [46] multiple ways to combine the human feedback processing of TAMER and classical Reinforcement Learning. In this paper, they assume that first a human trains a TAMER agent and subsequently the trained agent is used to improve the classical Reinforcement Learning agent. Based on

these methods, in [47] an approach for simultaneous training from the classical environmental reward and the human feedback is presented. There are two distinct models for the two kinds of information learned. Those can then be fused with multiple possible methods to decide for an action. To weight both models in the fusion, they present a method to measure how much a state action pair was already judged recently. If this value is high, the TAMER model is more trustworthy and thus should be weighted more.

2.2.3 Supervised Reinforcement Learning

One way to include human advice to the Reinforcement Learning process is the Supervised Reinforcement Learning method, presented in [40]. The idea is to include multiple channels of prior knowledge about the value of possible actions, e.g. a human advice for some action. For the Reinforcement Learning part, the original paper uses Q-learning ([48]).

First, all information channels are combined by adding them and normalizing the result. Then the compatibility of the given advice and the so far learned values is computed. If the compatibility is very low, it is assumed, that there is too little knowledge at the current state and the advice might be incorrect. Because of this exploration is performed, i.e. not advised actions are preferred. If this is not the case, the advice is combined with the learned values and a softmax policy is used to finally decide for an action.

One problem for applying this approach to human interaction is, that the human typically wants the agent to really perform what he suggested. Especially at the beginning of the training, the learned values are not yet meaningful. Here the advice might be especially helpful, but if it is by chance not compatible with the values, it is not used. Even worse, those actions that were the least recommended by the human are followed.

Thus, we use a simplified version, that only makes use of the principle idea: If an advice is given by the human, it is always followed by the agent. When there is no advice, the values learned by Reinforcement Learning are followed. For this Reinforcement Learning part, we use the SARSA algorithm ([43]).

3 Combination of Interactive Reinforcement Learning Algorithms

While there are many different approaches for Interactive Reinforcement Learning, we want to bring them into one principle formal framework. This shall lead to a better understanding of different methods and their differences. A diagram of the formal framework can be seen in Figure 3.1. We concentrate here on value based methods in the sense, that there is some model of value for different state action pairs. Nevertheless, those values do not necessarily correspond to the values of a classical Q-table. They just have to represent some notion of good and bad. We use this value based view because most existing work can be seen this way.

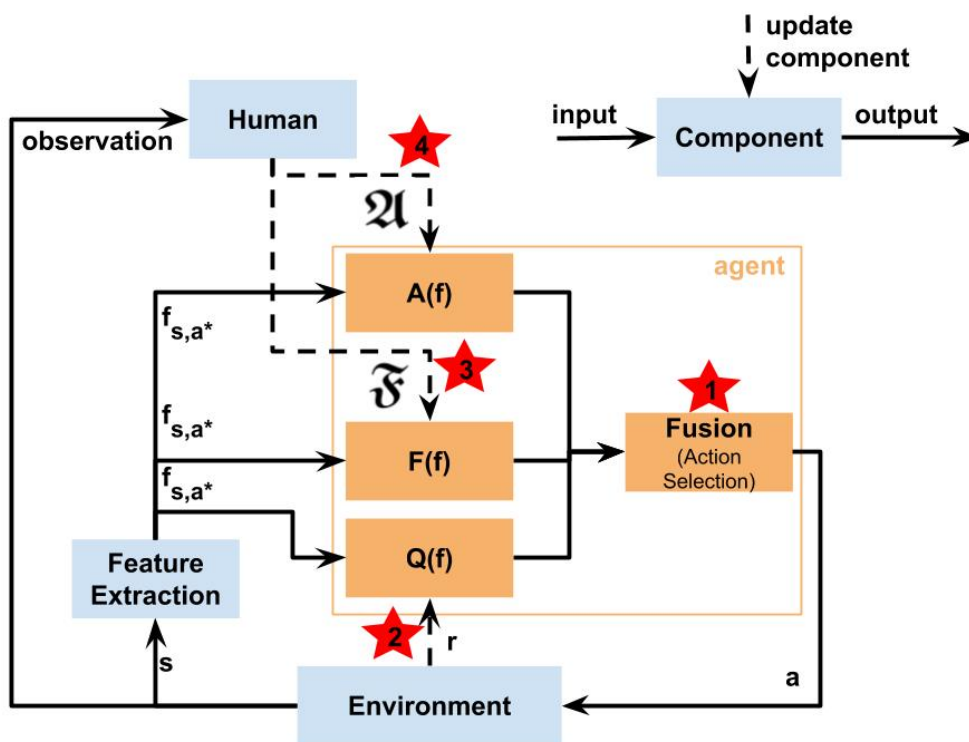


Figure 3.1.: Overview of the Proposed Framework

The human can observe the environment and give advice (\mathcal{A}) and / or feedback (\mathcal{F}). The agent learns one model (A, F, Q) for each present type of information: human advice, human feedback, and environmental reward. Those models are fused to decide on one action to perform next.

As already introduced, we assume that an Interactive Reinforcement Learning algorithm has some kind of a value model. More precisely, we assume up to three different models for different information:

Q: This model is learned from the environmental reward. Thus, classical Reinforcement Learning techniques can be used here.


F: This model is learned from feedback the human user gives for the behavior of the agent.

A: This model is learned from advice that the human user gives the agent for some state.

Those models can have an arbitrary representation. In this thesis, we use linear models, neural networks, and tables. If linear models or neural networks are used, their parameters are denoted by θ_Q, θ_F and θ_A respectively. The learned models can then be combined in different ways.

It is assumed, that there is some sufficient feature representation $\mathbf{f}_{s,a}$ of state s and action a given for each problem. For the implementation of the SARSA, the DeepTAMER and the DeepAdvice algorithms, the features only encode the state and a distinct model is learned for each action. Nevertheless, this can also be transferred to a single model with state action features (see Appendix A), we will keep the uniform notation.

3.1 Fusion of Learned Models

This part of fusing models and choosing actions is marked in Figure 3.1 by .

Assuming that at least one, but possibly multiple, models that represent some kind of value of state action pairs and corresponding weights for those models are given, an optimal action a' has to be chosen. There are multiple possible ways to do that. We use the Action Biasing and the Control Sharing techniques from [47], that we adopted to be able to handle human advice additionally to the environmental reward and the human feedback. However, if no human advice is used ($w_A = 0$), they behave exactly as described in [47].

Action Biasing (AB):

Here one model is computed as a linear combination of the given ones and is then used for the action selection.

$$a' = \operatorname{argmax}_{a \in \mathcal{A}} (w_Q Q(\mathbf{f}_{s,a}) + w_F F(\mathbf{f}_{s,a}) + w_A A(\mathbf{f}_{s,a}))$$

Control Sharing (CS):

This is a probabilistic approach that switches between the given models.

$$\begin{aligned} Pr(a' = \operatorname{argmax}_a Q(\mathbf{f}_{s,a})) &= 1 - \left(\min \left(w_F, \frac{w_F}{w_F + w_A} \right) + \min \left(w_A, \frac{w_A}{w_F + w_A} \right) \right) \\ Pr(a' = \operatorname{argmax}_a F(\mathbf{f}_{s,a})) &= \min \left(w_F, \frac{w_F}{w_F + w_A} \right) \\ Pr(a' = \operatorname{argmax}_a A(\mathbf{f}_{s,a})) &= \min \left(w_A, \frac{w_A}{w_F + w_A} \right) \end{aligned}$$

For this method, no weight for the Reinforcement Learning model is needed, because the corresponding model is simply used as default if the probabilities for following the F and A do not sum up to 1.

Choosing the used weights w_F, w_Q, w_A can then lead to different algorithms. Some examples are listed in the rest of this section. They can use both presented methods to fuse multiple models. And of course, other ways to fuse the models can be made up and used instead of Action Biasing and Control Sharing.

3.1.1 TAMER

Only the learned feedback model is used in the basic TAMER [12] algorithm:

$$w_Q = 0 \quad w_F = 1 \quad w_A = 0$$

This is the same for the DeepTAMER ([19]) algorithm. Because the Reinforcement Learning and the advice model are not used they do not need to be learned and these algorithms can learn from human feedback without any other information.

3.1.2 Advice

Only the learned model of advised actions is used in the Advice algorithm:

$$w_Q = 0 \quad w_F = 0 \quad w_A = 1$$

Again, this can be used the same in the deep version of the algorithm.

Although this algorithm only learns from actions that are suggested by humans, no complete demonstrations have to be given by the human. In contrast, the human trainer can observe the current behavior of the agent and modify it for specific states, where he thinks it is not optimal yet.

3.1.3 Supervised Reinforcement Learning (SRL)

In our Supervised Reinforcement Learning experiments (inspired by [40]) the advice is always used if provided. Otherwise, the RL model is followed. This can be expressed in our framework the following way:

$$\begin{aligned} w_Q &= 1 - w_A \\ w_F &= 0 \\ w_A &= \begin{cases} 1 & \text{if } \mathfrak{A} \text{ provided} \\ 0 & \text{else} \end{cases} \end{aligned}$$

For this algorithm, a reward function is needed and human advice is only used for exploration.

3.1.4 Simultaneous TAMER (Sim TAMER)

The simultaneous TAMER [47] algorithm combines human feedback and environmental reward. To do that, the weights are computed with the hyperparameters λ_{sim}, A and C .

ξ is inspired by eligibility traces and shall represent how much recent information about some feature is available. It has the same shape as the features, that are used for TAMER. ξ is initialized as zero vector. Then it is updated at each step:

$$\xi = \begin{cases} \min(\xi + A\mathbf{f}_{s,a}, \mathbf{1}) & \text{if } \mathfrak{F} \text{ provided} \\ \lambda_{sim}\xi & \text{else} \end{cases}$$

Based on this the weights can be computed:

$$\begin{aligned} w_Q &= 1 \\ w_F &= C\xi^T \mathbf{f} / \|\mathbf{f}\|_1 \\ w_A &= 0 \end{aligned}$$

Because the weight for the feedback model diminishes when no feedback was given for some time, the human feedback only supports the exploration here too. However, one interaction does not only affect a single action but potentially many ones.

3.1.5 Simultaneous Advice (Sim Advice)

The simultaneous Advice algorithm follows the same idea as the simultaneous TAMER:

$$\xi = \begin{cases} \min(\xi + A\mathbf{f}_{s,a}, \mathbf{1}) & \text{if } \mathfrak{A} \text{ provided} \\ \lambda_{sim}\xi & \text{else} \end{cases}$$

$$\begin{aligned} w_Q &= 1 \\ w_F &= 0 \\ w_A &= C\xi^T \mathbf{f} / \|\mathbf{f}\|_1 \end{aligned}$$

This shall enable a longer influence of a given advice, compared to the Supervised RL approach.

3.1.6 Simultaneous (Sim)

One option to combine all possible kinds of information in the proposed framework is the algorithm we will call Simultaneous and that is introduced in the following. It is supposed to combine Simultaneous TAMER and Simultaneous Advice with the hyperparameters C_F and C_A to weight both models:

$$pmbxi_F = \begin{cases} \min(\xi_F + A\mathbf{f}_{s,a}, \mathbf{1}) & \text{if } \mathfrak{F} \text{ provided} \\ \lambda_{sim}\xi_F & \text{else} \end{cases} \quad \xi_A = \begin{cases} \min(\xi_A + A\mathbf{f}_{s,a}, \mathbf{1}) & \text{if } \mathfrak{A} \text{ provided} \\ \lambda_{sim}\xi_A & \text{else} \end{cases}$$

$$\begin{aligned} w_Q &= 1 \\ w_F &= C_F \xi_F^T \mathbf{f} / \|\mathbf{f}\|_1 \\ w_A &= C_A \xi_A^T \mathbf{f} / \|\mathbf{f}\|_1 \end{aligned}$$

It can be seen that it is the simple combination of the formulas of Simultaneous TAMER and Simultaneous Advice.

3.1.7 Advised TAMER (Sup Advice)

To combine feedback and advice in the absent of an environmental reward function we again follow the idea of Simultaneous TAMER. However, in Simultaneous TAMER, w_F shall go to 0, if no feedback was given for a long time. But in the combination of Advice and TAMER, the weight should not decrease, because there is no default behavior given by a Reinforcement Learning model. Thus, we adjust the update of the traces and do not decrease them, even if no human information was provided in the last time step:

$$\xi_F = \begin{cases} \min(\xi_F + Af_{s,a}, \mathbf{1}) & \text{if } \mathfrak{F} \text{ provided} \\ \xi_F & \text{else} \end{cases} \quad \xi_A = \begin{cases} \min(\xi_A + Af_{s,a}, \mathbf{1}) & \text{if } \mathfrak{A} \text{ provided} \\ \xi_A & \text{else} \end{cases}$$

When computing the weights it has to be made sure that the Reinforcement Learning model is not used:

$$\begin{aligned} w_Q &= 0 \\ w_F &= \frac{C_F \xi_F^T \mathbf{f} / \|\mathbf{f}\|_1}{C_F \xi_F^T \mathbf{f} / \|\mathbf{f}\|_1 + C_A \xi_A^T \mathbf{f} / \|\mathbf{f}\|_1} \\ w_A &= \frac{C_A \xi_A^T \mathbf{f} / \|\mathbf{f}\|_1}{C_F \xi_F^T \mathbf{f} / \|\mathbf{f}\|_1 + C_A \xi_A^T \mathbf{f} / \|\mathbf{f}\|_1} \end{aligned}$$

For the Control Sharing fusion method, it is necessary that the weights for human interaction add up to one ($w_F + w_A = 1$) to avoid the Reinforcement Learning model. Thus they are normalized but stay the same in their ratio.

3.1.8 Supervised Advice

This algorithm is an exception in this list of example algorithms because it combines two Advice models. One of the models is updated as in the Advice algorithm and will be called A_A here. The other model is updated as in Supervised Reinforcement Learning and will be called A_S .

The idea of this algorithm is, to learn a model as in the Advice algorithm, but make sure that, if the advice was given at the current time step, it is always followed. This algorithm can be seen as a modification of the Supervised Reinforcement Learning algorithm, where the Reinforcement Learning model is replaced by the Advice model. Thus, the combination of the models is similar to Supervised Reinforcement Learning:


$$\begin{aligned} w_Q &= 0 \\ w_F &= 0 \\ w_{A_A} &= 1 - w_{A_S} \\ w_{A_S} &= \begin{cases} 1 & \text{if } \mathfrak{A} \text{ provided} \\ 0 & \text{else} \end{cases} \end{aligned}$$

In principle, the framework can be extended by arbitrary many additional models. This algorithm can be seen as an example how the framework can be extended. In general, the fusion method would have to be adapted, too, to be able to handle the additional model. In this example, this is trivial, because there is always one weight that is 1, while all other weights are 0. Thus, the model whose weight is 1 is always followed.

3.2 Model Updates

To be able to combine the different models, they have to be learned first. In this section, the methods to update the models for possible kinds of information are presented. They include linear as well as deep models.

3.2.1 $Q(f)$ - Reinforcement Learning Update

The model of the environmental reward is learned with Reinforcement Learning. We will use either SARSA or DQN, depending on the necessary model complexity. In Figure 3.1 the Reinforcement Learning model update, based on environmental reward, is marked by .

The error that shall be minimized in **SARSA** is

$$\epsilon = r + \gamma \max_{a'} Q(\mathbf{f}_{s',a'}) - Q(\mathbf{f}_{s,a}),$$

where action a was performed in state s and led to s' . a' is the result of action selection in state s' . Eligibility traces are computed by

$$\xi = \gamma \lambda \xi + \nabla_{\theta_Q} Q(\mathbf{f}_{s,a}).$$

They can then be used to update the model:

$$\theta_Q = \theta_Q + \alpha \epsilon \xi$$

If **DQN** is used, there are two models present. The one that is used for the action selection will still be called Q , but additionally, there is the so called target network that will be denoted by Q' here. Instead of doing one model update for each performed step in SARSA, all data that is acquired during one step is saved in a replay memory. After each step, multiple data points are sampled from the memory and used for the update. We assume that the sampled batch b from the replay memory has N entries, each consisting of state, action, next state, reward, done in this order:

$$b = \{(s_i, a_i, s'_i, r_i, d_i); i = 1, \dots, N\}$$

Based on a sampled batch, a target value is computed:

$$t_i = r_i + \gamma d_i Q'(s'_i, \operatorname{argmax}_{a'} Q(s'_i, a'))$$

This target is compared with the current estimate by the mean squared error:

$$\epsilon = \frac{1}{N} \sum_{i=1}^N (Q(s_i, a_i) - t_i)^2$$

This error is used for the gradient update step:

$$\theta_Q = \theta_Q - \alpha \nabla_{\theta_Q} \epsilon$$

In a regular frequency, the target network is updated by the current estimate: $Q' = Q$

3.2.2 $F(f)$ - TAMER Update

The first form of human information that is included here is feedback. In Figure 3.1, the feedback model update is marked by \star . It can be implemented with the method of the **TAMER** algorithm.

For $i = 0, 1, 2, 3, \dots$ assume the action a_i was performed in state s_i at time t_i and that the weight w_i tells how much s_i, a_i influenced the given feedback \mathfrak{F} . The time is assumed to be 0, when the feedback was given and positive before.

To compute the weights a uniform distribution over time is used. Humans need some time to react to perceptions. Here some fixed \min_{react} is assumed to be this time the human needs to react. Besides, it is assumed, that the human does not judge actions, that are older than \max_{react} . Thus, the uniform distribution $P(t) = \text{Uniform}(\min_{\text{react}}, \max_{\text{react}})$ is used to assign credit for feedback to actions. So, the weights are computed as

$$w_i = \int_{t_{i-1}}^{t_i} P(t) dt.$$

To minimize the error

$$\epsilon = \mathfrak{F} - F\left(\sum_i w_i \mathbf{f}_{s_i, a_i}\right),$$

the following update to the model is performed:

$$\theta_F = \theta_F - \alpha \epsilon \nabla_{\theta_F} \epsilon$$

Alternatively, the **DeepTAMER** algorithm can be used, if the feedback model shall be represented by a neural network. In this case, the weights w_i are computed the same, but a mean squared error is used as loss for all steps that have a weight greater 0 $I = i; w_i > 0$:


$$\epsilon = \frac{1}{|I|} \sum_{i \in I} w_i (\mathfrak{F} - F(\mathbf{f}_{s_i, a_i}))^2$$

The error is used for a standard gradient update step:

$$\theta_F = \theta_F - \alpha \nabla_{\theta_F} \epsilon$$

Besides, for all $w_i > 0$ the corresponding data is saved in a replay memory. When no feedback was given in a step and the memory is already big enough, a batch is sampled from it and used to update the model. This is done the same way as described above. The only difference is, that there is not a single feedback value \mathfrak{F} but a batch of data sampled randomly from different times feedback was given is used.

3.2.3 $A(f)$

The other form of human information that is treated in this thesis is advice, marked by  in Figure 3.1. To directly perform the suggested action, a tabular representation for A is used as follows for the **Supervised Reinforcement Learning**:

$$A(f_{s,a}) = \begin{cases} 1 & \text{if } a == \mathfrak{A} \\ 0 & \text{else} \end{cases}$$

This way, the human information is used only once. A natural alternative would be to also learn a model here, to be able to reuse the provided information. Thus, we also use learning a linear model as an alternative. This model is used for the **Advice** algorithm. We assume, that an advised action should have value 1, while all other actions get a value of 0:

$$T = \begin{cases} 1 & \text{if } a == \mathfrak{A} \\ 0 & \text{else} \end{cases}$$

With this assumption the mean squared error can be computed:

$$\epsilon = \frac{1}{|\mathcal{A}|} \sum_{i=1}^{|\mathcal{A}|} (T - A(\mathbf{f}_{s_i}))^2$$

This error can then be used to update the model:

$$\boldsymbol{\theta}_A = \boldsymbol{\theta}_A + \alpha \nabla_{\boldsymbol{\theta}_A} \epsilon$$

Instead of a linear model, a neural network can be used as well with the **DeepAdvice** algorithm. The only difference is a replay memory, where all experienced state target pairs are saved and that is used for additional updates. The loss is computed as mean squared error over a batch of size N from the memory:

$$\epsilon = \frac{1}{|\mathcal{A}|N} \sum_{j=1}^N \sum_{i=1}^{|\mathcal{A}|} (T_j - A(\mathbf{f}_{s_j, a_i}))^2$$

where T_j and s_j with $j = 1, 2, 3, \dots, N$ are the sampled targets and corresponding states. This error can then be used for the same update as described for the linear model.

4 Implementation Details

There are two main parts that had to be implemented. On the one hand, there are the used algorithms, and on the other hand there is the interaction with the human. Both parts can also be seen in Figure 4.1.

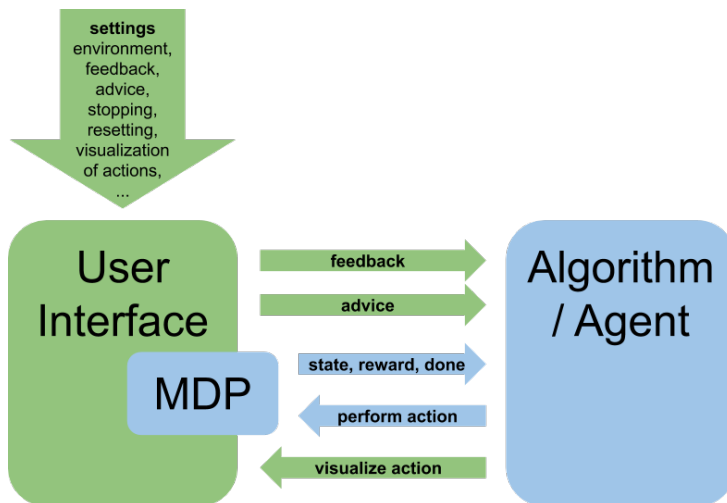


Figure 4.1.: Structure of the Code

The code can be structured in two main parts. On the one hand there is the Interactive Reinforcement Learning algorithm. The algorithm can also be seen as the agent, because it observes the environment and chooses the actions that shall be performed. On the other hand, there is the user interface that includes the environment that shall be learned. The whole user interface can be seen as a kind of environment that provides feedback and / or advice as additional information, compared to a classical Reinforcement Learning environment. The two parts run in parallel and only communicate over several queues. In this figure, all arrows represent a queue that is used for communication. The settings queue is only used previous to a training run, to specify what exactly shall be done.

In the following, both parts are shortly explained. The part explained in the Algorithms section can be used independently from the User Interface.

4.1 Learning Algorithms

All used algorithms, including the Reinforcement Learning baselines, implement an abstract *Algorithm* class. They all must have a reward model, that can either be linear or a neural network. Other model implementations could easily be added. This model can then be used by an ϵ -greedy policy. Next to the `__init__` method, all algorithms must implement a `reset` and a `learn` method. The training results can be evaluated with the `eval_policy` method. An example for the `learn` method can be seen in Listing 4.1, where its implementation of the TAMER algorithm is shown. This method is also used by the DeepTAMER implementation.

```
def learn(self, episodes=None):
    """
    Start the training
    """
    s = self.env.reset()
    if self.render:
        self.env.render()
    iteration = 0
    episode = 0
    episodic_reward = [0]
    if not episodes:
        episodes = self.num_episodes
    while episode < episodes:
        iteration += 1

        action_time = time.time()
        self.send_action(s)
```

```

f = self.features.transform(s, self.proposed_a)
self.crediter.update_features(f, action_time)

last_s = s
try:
    s, r, terminal, _ = self.env.step(self.proposed_a)
except KeyboardInterrupt: # The training was quited by the user
    break
except ResetException: # The current episode was reset by the user
    episode += 1
    episodic_reward[-1] = None
    episodic_reward += [0]
    s = self.env.reset()
    continue
if self.render:
    self.env.render()

h, t = self.get_feedback(last_s, self.proposed_a)
self.crediter.update_reward_time(t)
if h is not None:
    self.feedback_accu[episode, int(h > 0)] += 1

error = self.compute_error(h, iteration)

if error != 0:
    self.reward_model.update(error)

episodic_reward[-1] += r
if terminal:
    episode += 1
    episodic_reward += [0]
    if episode < episodes:
        try:
            s = self.env.reset()
        except KeyboardInterrupt:
            break
    self.eps -= self.eps_decay

return np.array([episodic_reward[: -1]])

```

Listing 4.1: *learn* Method of TAMER Algorithm

This method shows the basic operations of the TAMER algorithm. All implemented algorithms have a *learn* method. They are all structured similarly, so this can be seen as an example for all *learn* functions.

In Listing 4.2 can be seen the code that is used by the algorithms to get advice. This way it is possible to use the algorithms, independently of the use of human or oracle information.

```

def get_advice(self, s):
    """
    Get advice either from generator or queue.
    None, if no advice is provided
    :param s: state
    :return: advice, the state for that the advice was given
    """
    if self.advice_generator is not None:
        advice = self.advice_generator.get_advice(s)
        advised_state = s
    elif self.advice_queue is not None:

```

```

advice, advised_state = None, None
while not self.advice_queue.empty():
    advice, advised_state = self.advice_queue.get(0)
else:
    advice, advised_state = None, None
return advice, advised_state

```

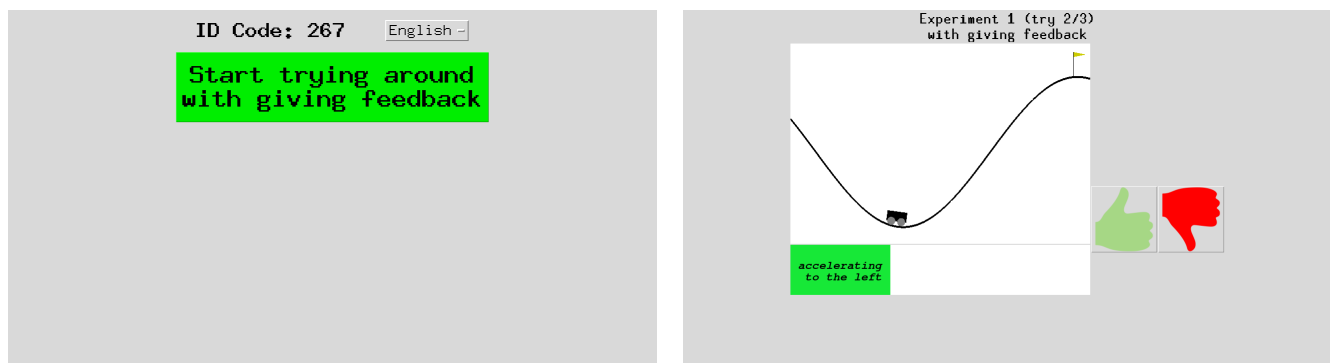
Listing 4.2: Code to Get Advice

To include human interaction this method and an equivalent for feedback are used. The information can either come from an oracle or the interaction with a real user. With this method this difference is abstracted.

For experiments with oracle information, those oracles implement the abstract *FeedbackGenerator* or respectively *AdviceGenerator* class. Openai gym¹ environments or other environments that implement its *Env* class can be used. With matching features that implement the abstract *Features* class, this is already sufficient for experiments with oracle human interaction. For the tile coding feature extraction, code from the mushroom Reinforcement Learning python library² is used.

4.2 User Interface

When experiments with real human users shall be rolled out, the second part of the implementation is needed: the user interface.



(a) Start Frame

(b) Run Frame

Appearance of the user interface before the training run was started. Appearance of the user interface during the training.

Figure 4.2.: User Interface

An example how the user interface can look like with the TAMER algorithm and the Mountain Car environment.

In the TAMER algorithm, credit for feedback is assigned to actions based on the time that passed between the action execution and the human interaction. To get the correct time, when exactly the human interacted with the system, the input of feedback has to be in a different thread than the actual algorithm. Based on this necessity we structured the code in two parts that run in parallel and only communicate over queues: the human interaction and the actual algorithm. Another advantage that comes from this distinction is that the computations of the algorithm and visualizations for the user interaction can be computed in parallel, what allows a fluent run of the experiments.

This structure follows the intuition that the human can be seen as part of the environment. To keep the interaction with the environment simple for the algorithm, there is a *InteractionEnv* class that has the same methods as a gym environment and handles the communication with the user interface. The actual environments must have a variable *step_wait* that contains the time that shall pass between two executions of actions in milliseconds. If the number of allowed interactions shall be limited, this has to be specified in the environment's variable *interaction_limit*. Besides, it is required that the *rgb_array* mode for *render* method is implemented and that rendering in this mode does not open any windows. Thus, the gym environments can not directly be used for experiments with real humans. The used gridworld implementations are based on code from <https://github.com/xinleipan/gym-gridworld>.

¹ <https://github.com/openai/gym>

² <https://github.com/AIRLab-POLIMI/mushroom>

To incorporate human advice and feedback, the algorithms have methods `get_advice` or `get_feedback`, respectively. They check which kind of information is available, an oracle or a queue to get information from a real human. The `get_advice` method can be seen in Listing 4.2. It is possible to use buttons or the keyboard for the interaction. To enable this, the relation between actions keys and images that are displayed on the buttons have to be specified.

In the user interface, there is a dropdown menu to select the language that is used to display texts. There is a file that contains all strings that can be displayed. To support an additional language to the existing ones, German and English, a new entry for the language can be added.

The user interface is implemented with the `tkinter` package. It is switched back and forth between one frame that is shown before a training run and the frame that is used for the actual training. An example for each frame can be seen in Figure 4.2.

```
def check_setting_q(self):
    """
    Check, whether there are new settings
    :return: whether there are new settings
    """
    try:
        settings = self.setting_q.get(False)

        self.run_frame.forget()

        if settings == "Finish":
            self.finish_frame.pack()
            self.finish_frame.show()
            return True

        self.feedback, self.advice, self.vis_actions, self.stoppable, \
            self.resettable, env, info = settings
        self.env = my_envs.get(env)()
        self.env.reset()
        self.phase, self.number_episodes, self.algorithm = info
        self.finished_episodes = 0
        if self.phase == "train":
            self.finished_trainings += 1
        else:
            self.finished_trainings = 0

        self.start_frame.update_start_button()
        self.start_frame.pack()

        self.setting_q.task_done()
        return True
    except Empty:
        return False
```

Listing 4.3: Checking the Settings Queue in the User Interface

The setting queue is checked regularly. If information is provided, the current training run is stopped and the frame from which the next training run can be started is shown.

The queues that forward information to the user interface are regularly checked. As an example for this, in Listing 4.3 can be seen how the queue, that contains the settings for a new training run is, checked. If new settings are available, the displayed frame is switched. As can be seen in the code snippet, things like the used algorithm, the use of advice, feedback, or both, the used environment and whether an episode can be reset or the training can be ended by the user, are communicated over this queue. This way, all listed properties can be changed during one user study.

5 Experimental Evaluation

To compare different approaches, they were applied to multiple environments in experiments with simulated interaction and real humans. To be able to perform experiments without the need for interaction with real humans, oracles for automatic advice and feedback generation were implemented for a part of the environments. The simulated human interaction is helpful to run many experiments with different algorithms, environments, and settings. This way it can help to find suitable hyperparameters. Besides, the achieved results are comparable and there is no noise, that would be introduced when dealing with real humans. On the other hand, it is not clear whether humans really act exactly the same way as our oracle implementations do - probably they do not. Because of that, it is necessary to perform experiments with real humans. This is essential because the whole area of Interactive Reinforcement Learning is about enabling humans to train agents. If they don't work with real humans, the algorithms miss their target.

The experimental evaluation is structured as follows: First, the used environments are described. Then, the results of the different algorithms in Simulation are presented. Finally, the experiments with real humans, called user studies, are explained and the results are presented.

5.1 Learning Environments

To evaluate the presented algorithms, two gridworlds, the openai gym Mountain Car, and the openai gym Lunar Lander are used. They were chosen to evaluate different properties. The gridworlds are very easy to understand by a human trainer. Besides, the human can clearly assign his information to one specific time step. Thus, they represent cases where the human trainers should be able to really give useful information.

In contrast to this, the Mountain Car is harder to understand and has a high action frequency. Due to this, it is harder for a human to give information. On the other hand, there are only two actions that are important to solve the task.

The Lunar Lander combines four relevant actions with a high action frequency and a, compared with the other environments, very complex behavior of the agent.

5.1.1 Gridworld



Figure 5.1.: The Gridworld Environment

A visualization of the grid can be seen in Figure 5.1. The start position is sampled randomly from the green area at the begin of each episode. The agent has to reach one of the three goals. There is one goal that is denoted by a single coin and yields a reward of 1. Another one of them is marked by multiple coins and gives a reward of 10. Finally, there is

one goal that is marked by a treasure chest and yields a reward of 50. You can also see this in the legend in Figure 5.1, where the rewards of the different goals are given in parentheses. For each step that does not reach one of the goals, a reward of -1 is provided. The black fields are walls, that the agent can not pass. The current position of the agent itself is marked by the yellow smiley. One episode consists of 200 steps if no goal was reached before.

The oracle for the grid world is implemented the following way:

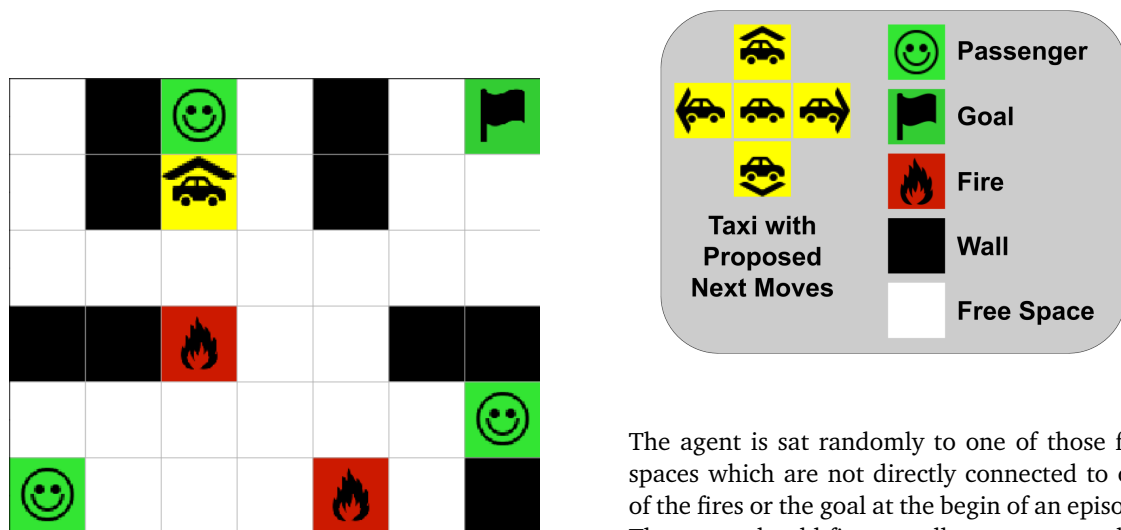
- Feedback: All actions that reduce the agent’s distance to the nearest goal get a reward of 1. All other actions get a reward of -1 .
- Advice: There is one prescribed optimal action for each state, that is always used.

At each time step, there is a probability of 50% that feedback/advice is given for all experiments.

The grid is a relatively simple problem for humans so that it should be easy for humans to provide feedback or advice. On the other hand, a Reinforcement Learning agent needs considerable exploration to find at least one of the goals. Thus, we considered it to be well suited to point out some advantages of Interactive Reinforcement Learning.

5.1.2 Taxi like Gridworld with Holes

This environment is similar to the one presented in [34]. Nevertheless, we introduced some differences. For a visualization see Figure 5.2.



The agent is sat randomly to one of those free spaces which are not directly connected to one of the fires or the goal at the begin of an episode. The agent should first get all passengers and go to the goal afterwards. He should never run into one of the fires.

Figure 5.2.: The Taxi Environment

One difference to the original environment are the fires we introduced. They simulate some catastrophic behavior, so we can show how humans handle that and whether humans are helpful in such situations. If the agent enters a fire field, the episode ends.

In the original Taxi environment, each step is performed in a random direction with a probability of 10%. Those random steps might confuse the human trainer. Because of that, we only use deterministic steps. On the other hand, we introduce randomness in the start state. In the version of [34] the agent always starts in the top left corner of the world. In contrast to that, the agent starts randomly from some free field. All free fields except those that are directly next to either a fire or the goal are possible starting positions. This way it is avoided that the very first step can end an episode. If an episode is not ended by the goal or a fire, it consists of 200 steps.

If the agent steps at a passenger field, he automatically picks up the passenger, who is no longer visualized for the rest of the episode. Stepping at the goal or a fire field immediately ends an episode. The rewards are distributed as follows: Entering a fire is punished with a reward of -15 . When the agent reaches the goal, the reward depends on the number of passengers he has picked up before. If no passengers were picked up, there is a reward of 0, one passenger leads to a reward of 1, two passengers are worth a reward of 3, and three passengers yield a reward of 15. All other steps are rewarded by 0.

The oracles are implemented similarly as for the gridworld before:

- Feedback: All actions that reduce the agent’s distance to the nearest passenger get a reward of 1. If all passengers were picked up already, the actions that bring the agent nearer to the goal are rewarded with 1. All other actions get a reward of -1 .
- Advice: There is one prescribed optimal action for each state, that is always used.

As in the last gridworld there is a probability of 50% at each time step that feedback or advice is given. For the visualization in the user study, the action that the agent currently plans to perform next is displayed by an arrow. This is explained in more detail in the description of the corresponding user study in Section 5.3.

5.1.3 Mountain Car

The two gridworlds, presented before, were naturally rather similar. With the Mountain Car¹ from the openai gym another environmental type with a continuous state space and a high action frequency is introduced. In this environment, whose visualization can be seen in Figure 5.3, the goal is to reach the flag on the right hill as fast as possible. The car has to swing up on the left hill first, to get enough momentum, to get up the whole right hill. One episode consists of 200 time steps. Each step yields a reward of -1 .

In this environment, it is harder for humans to provide advice or feedback due to two reasons. First, it is a real-time problem with a high frequency of actions and the user thus has to react fast enough. One time step takes about 0.068 seconds in the user studies performed in this thesis. Second, it is not always clear for humans what the optimal action is.

The oracles also do not provide perfect feedback and advice for this environment: Accelerating in the direction of the current movement of the agent gets positive feedback of 1 and is advised. Not accelerating or going in the opposite direction of the current movement gets a reward of -1 . At each time step oracle interaction is provided with a probability of 5%. The interaction frequency was reduced, compared to the gridworlds, because the actions are performed with a higher frequency and thus a real human can not judge each action individually.

5.1.4 Lunar Lander

The most complex environment used in this thesis is the openai gym Lunar Lander². The goal of this environment is to safely land between two displayed flags on the moon surface. It is especially good to do this fast and with only a little consumption of fuel. For the detailed reward function, see the implementation at https://github.com/openai/gym/blob/master/gym/envs/box2d/lunar_lander.py. However, the main goal is to bring the lander to the moon without a crash.

One episode consists of at most 1000 time steps, if none of the following events, that immediately end an episode, occurs: The lander reaches successfully the moon; the lander crashes down to the moon; the lander leaves the screen on the left or right side. In the experiments with real humans, one time step takes about 0.063 seconds.

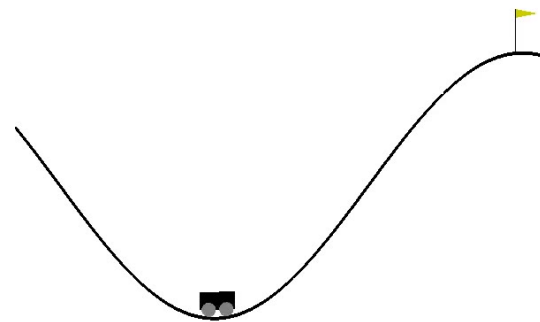


Figure 5.3.: The Mountain Car Environment
The car has to reach the flag on the right hill as fast as possible

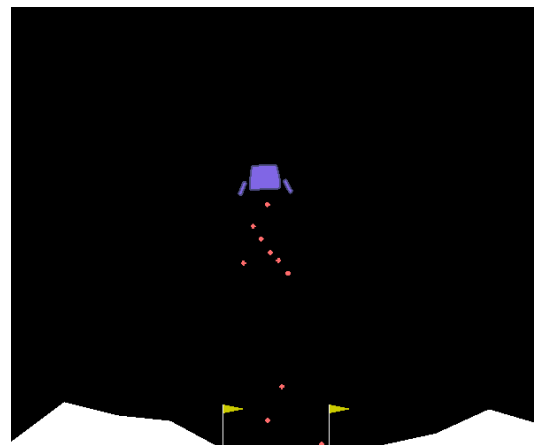


Figure 5.4.: The Lunar Lander Environment
The lander has to reach the surface of the moon safely. Optimally he reaches the area between the two flags fast and with little consumption of fuel. The visualization was extended at the top.

¹ <https://github.com/openai/gym/wiki/MountainCar-v0>

² <https://gym.openai.com/envs/LunarLander-v2/>

The visualization of the environment was extended to the top in comparison to the original implementation. This way the behavior of the lander in this area can also be judged by the human. This is not unfair to learning from environmental reward, because the state representation and reward function is also not limited to the top of the visualization but works the same for all heights.

Providing feedback or advice in this environment is even harder than for Mountain Car because more complex movements are possible. To solve the Mountain Car environment, only two actions are needed, while there are four different actions that are all useful for the Lunar Lander. Besides, position and velocity are only one dimensional for the Mountain Car and two dimensional for the Lunar Lander. Additionally, the orientation of the lander is important to successfully reach the moon.

We did not implement an oracle for advice or feedback for this environment, because it was not clear what might be a plausible human behavior.

5.2 Simulation Results

In this section, the feedback and the advice are automatically generated by an oracle. This enables comparable results for many different experiments and settings. The way how advice and feedback are computed is explained in the description of the used environment in Section 5.1.

All experiments presented in this section were performed with 20 training runs. The plots of this section were created with the following properties: All learning curves shown in this section with simulated human interaction show the mean episodic reward. The shadow of the curves is the 95% confidence interval under the assumption of Gaussian noise. For the final behavior, the boxplots of the episodic reward are created following this scheme: The box extends from the lower to upper quartile values of the data, with a line at the median. The whiskers extend from the box to show the range of the data. Flier points are those past the end of the whiskers. To get the data for the final behavior, all learned models were each evaluated for 25 episodes. During this evaluation, no feedback or advice was provided and no model updates were performed.

5.2.1 Gridworld

For the first grid environment, it is possible to learn an optimal policy without a reward function and only with the oracle information. Thus, in Figure 5.5 the environmental reward is only used for the SARSA baseline and the Supervised Reinforcement Learning (SRL) that always requires environmental reward additionally to the advice. The interactive

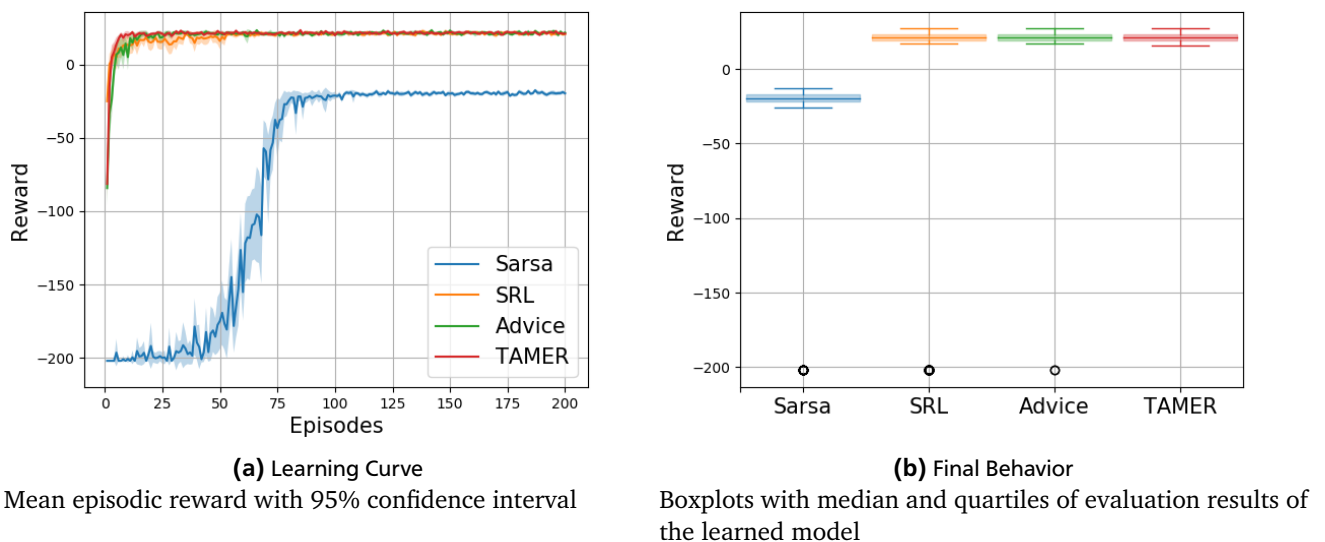


Figure 5.5.: Simple Gridworld Experiments

It can be seen that the agent can reach the highest goal after a short training time when information from the oracle is available. The Reinforcement Learning baseline takes much longer and only learns to find the lower goals.

algorithms can learn much faster because the human guides them to the goal. This reduces the amount of necessary

exploration. Besides, the SARSA baseline gets stuck at the lower rewarded goals and is not able to find the optimal goal. Because of this, its final behavior is worse than the one of the interactive algorithms. Furthermore, it can be seen, that TAMER is the only algorithm, that completely avoids to not reach a goal (reward of -200) in the evaluation. With the model learned with Advice this only happens in 0.4% of the evaluation results. SRL apparently worsens the behavior in this regard with 8.8% of the evaluation episodes not reaching any goal compared with 3% of the SARSA baseline.

	SARSA	SRL	Advice	TAMER
features	one-hot	one-hot	RBFs	RBFs
learning rate	0.1	0.1	0.01	0.1

Table 5.1.: Settings for the Experiments on the Gridworld Environment

It turned out that the models of oracle information performed better with RBF features, while the models of environmental reward did profit from a one-hot encoding. The used learning rate is the best that was found in a grid search.

For SARSA and SRL hyperparameters were $\lambda = 0.9, \gamma = 0.9$. The SRL model was initialized, such that all values were equal to -50 to ensure that suggested actions are not avoided at the beginning of the training. For the action selection in all algorithms, a greedy policy was applied to the learned models. For TAMER, feedback was only applied to the last performed action.

5.2.2 Taxi like Gridworld with Holes

For this gridworld, we compare the results of simple TAMER and Advice with the SARSA Reinforcement Learning baseline. A visualization of the results can be seen in Figure 5.6. Due to the different structure of the reward function, there is only a relatively small set of possible outcomes. Thus, we decided to use a histogram instead of boxplots.

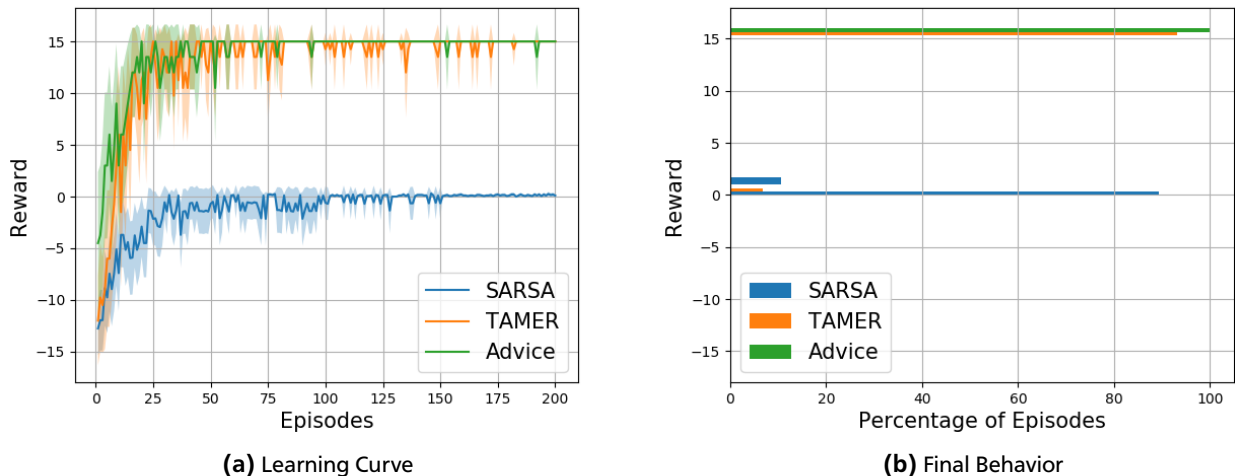


Figure 5.6.: Taxi like Gridworld with Holes Experiments

It can be seen that the agent can deliver most of the time all passengers to the goal after the training with oracle information. The Reinforcement Learning baseline only learns to bring at most one passenger to the goal, while it approaches the goal directly in most of the runs.

As can be seen in Figure 5.6 we were not able to learn a good epsilon greedy policy with SARSA. This corresponds with the results of [49]. The learned models were only able to bring at most one passenger to the goal. In contrast to that, the interactive algorithms can learn the task quite fast. With the use of Advice, all evaluation episodes led to the optimal reward of 15. The models learned with TAMER could also achieve this in most of the cases. All algorithms, including the Reinforcement Learning baseline, learn to always avoid fires.

The hyperparameters are chosen similarly as in the experiments with the first grid. For the SARSA baseline, a one-hot encoding was used as features and for the interactive algorithms, RBF features are used. The learning rates are 0.1 for

SARSA and 0.01 for the algorithms that are based on information from the oracles. For SARSA, some randomness was necessary. We used an ϵ -greedy policy that linearly reduced ϵ from 0.1 at the beginning to 0 at the end. We tried also to use more randomness, but this just worsened the results. The hyperparameters $\gamma = 0.9$ and $\lambda = 0.9$ were just kept as in the other gridworld experiments.

5.2.3 Mountain Car

Next to the grid environments, we also made experiments with the Mountain Car environment. When learning only from the oracle information, the desired behavior can be achieved quite fast again (see Figure 5.7). So, assuming that the simulated interaction is similar to human interaction, the behavior of driving up the hill can be taught to the agent by humans. However, the oracle does not provide optimal information. Thus, the algorithms do not reach the final performance of the SARSA baseline. The learning curves, as well as the final behavior, appear similar for all algorithms that learn only from oracle information.

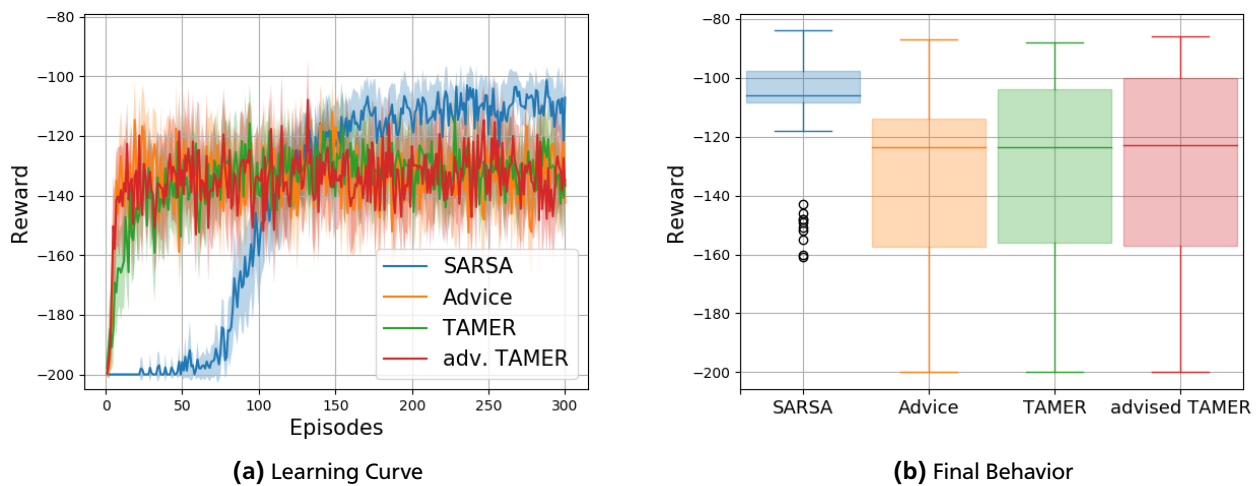


Figure 5.7.: Mountain Car Experiments without Environmental Reward

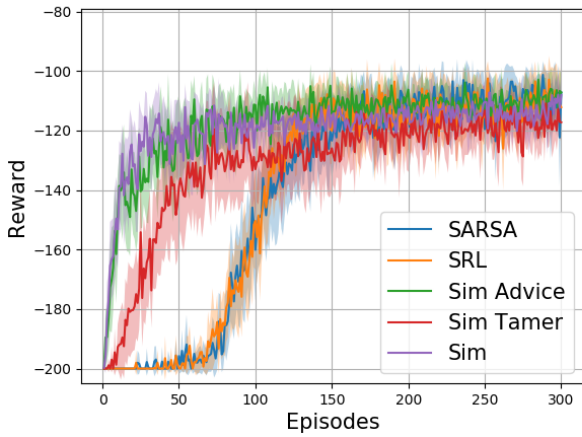
With information from the oracle, an episodic reward of about -135 can be reached very quickly. However, in the long term, the Reinforcement Learning baseline outperforms the algorithms without environmental reward, clearly.

Algorithm	learning rate
SARSA	0.01
SRL	0.01
Advice	0.1
TAMER	0.1
Advised TAMER	Advice: 0.1, TAMER: 0.05
Simultaneous Advice	SARSA: 0.01, Advice: 0.1
Simultaneous TAMER	SARSA: 0.01, TAMER: 0.05
Simultaneous	Advice: 0.1, TAMER: 0.05

Table 5.2.: Learning Rates for Experiments on the Mountain Car Environment

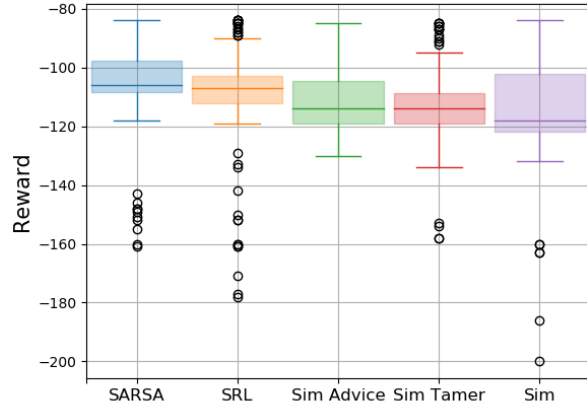
Different learning rates were used to avoid that one algorithm was disadvantaged by a bad setting. The best learning rates that were found in a small grid search were used. In the algorithms that combine different kinds of information, multiple models were learned, and thus there is one learning rate for each of the models.

To improve the results, we can assume that the reward function is available and use it with the algorithms that combine human information and the environmental reward. The results of those combinations can be seen in Figure 5.8. In this case, Simultaneous Advice (Sim Advice) and the combination of Advice, TAMER, and SARSA (Sim) learn the fastest. The Simultaneous TAMER can also accelerate the training compared to SARSA, but not quite as much. With Supervised



(a) Learning Curve

Mean episodic reward with 95% confidence interval



(b) Final Behavior

Boxplots with median and quartiles of evaluation results of the learned model

Figure 5.8.: Mountain Car Experiments with Environmental Reward

The combination of environmental reward with the oracle information can speed up the training with most algorithms. However, they still do not fully reach the Reinforcement Learning baseline in the evaluation.

Reinforcement Learning can no substantial improvement be achieved. This is because, in our implementation of the advice oracle, actions are only sparsely advised. This switching between the learned Q-model and the currently given advice can not improve the learning. In contrast to that, in the Simultaneous Advice algorithm, the given advice can be followed for multiple actions, because there is a learned model. This problem is also plausible for real human interactions because it is hard, or even impossible, for humans to give advice for each action if the action rate is high in real-time problems. For the final behavior, it can be seen in Figure 5.8 that the interactive algorithms are again worse than the SARSA baseline. However, this is now a minor difference compared to the results without environmental reward.

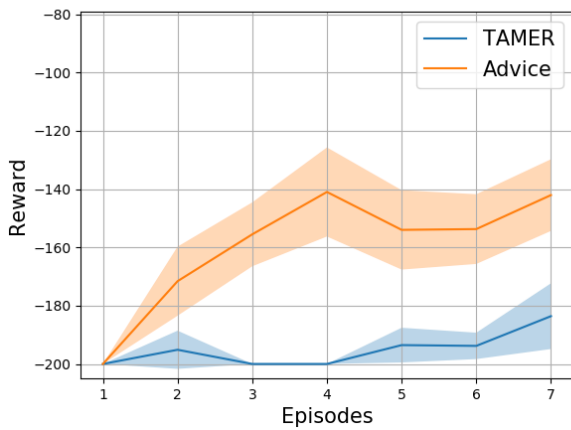
Again for Supervised RL and SARSA, also if it was part of some combined method, tile coding was used for the feature extraction, while all others used RBFs. For algorithms that include SARSA, hyperparameters were $\gamma = 1$ and $\lambda = 0.9$. For the action selection in all algorithms, a greedy policy was applied to the learned models. For TAMER, also if it was part of some combined method, the feedback was equally applied to the last 5 actions. For all combination algorithms, Action Biasing was used for the fusion of different models. Control Sharing did not work and thus was not included here. This is due to its property that it leads to the flickering of different actions, similarly to the Supervised Reinforcement Learning. For the Mountain Car, however, it is important to keep one action for some time to gain momentum that is needed to climb up the hill. The hyperparameters were set to $\lambda_{sim} = 0.9, A = 0.7, C_F = C_A = 50$. Besides, their SARSA models were initialized, such that all state action pairs had a value of about -120 because this is near to an optimal episodic reward. Differences and influences of the initialization of the reward models are also described in the original Simultaneous TAMER paper [47].

5.3 User Studies

To see how the algorithms perform in the interaction with real humans, multiple user studies were performed. For each one, we created a form with some basic questions and an explanation, what the participants should do. This shall make sure that each participant has the same documented knowledge. The full forms can be seen in Appendix B. For user studies, real humans are needed. Those can only take part once, per used environment, because they gain task knowledge during the user study. Thus, taking part in several user studies in the same environment would bias the result. Because of this, the amount of experiments that can be performed is limited.

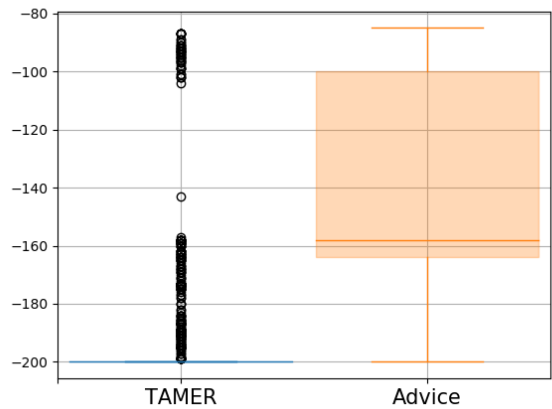
5.3.1 Mountain Car (1)

The first user study focuses on the principal difference between giving advice and giving feedback. To do this, we used the Mountain Car environment. For giving advice, the Advice algorithm was used, for feedback, the TAMER algorithm. With this user study we want to answer the following questions:



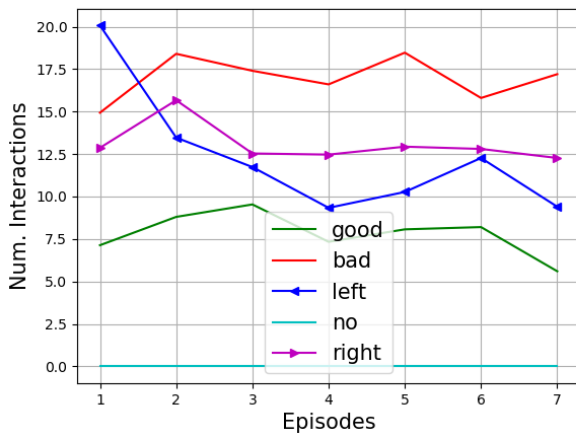
(a) Learning Curve

Mean episodic reward with 80% confidence interval



(b) Final Behavior of the Trained Models

Boxplots with median and quartiles of evaluation results of the learned model



(c) Average Number of Interactions

Here the amount of human feedback (good, bad) and advice (left, no, right) that was given during the user study can be seen.

Figure 5.9.: Results of the First User Study

- **Can the behavior** - that the car drives up to the flag - **be taught** using the TAMER and the Advice algorithm? Does the car reach the flag within 200 time steps following the learned model without interaction with a human in at least 80% of the trials?
- Does one of the two algorithms lead to a **better behavior during the training**? Perform a Wilcoxon test on the null hypothesis that both algorithms lead to the same cumulative reward during the training.
- Does one of the two algorithms lead to **more interactions** (giving advice or feedback)? Perform a Wilcoxon test on the null hypothesis that there are equally many interactions with both algorithms.

For the Wilcoxon tests, a level of significance of 0.01 is used. We decided on the Wilcoxon test because we want to verify, whether the central tendencies differ significantly. Besides, each participant used both algorithms three times, and that is why the results can be seen as paired (e.g. the first try with TAMER matches the first try with Advice of the same participant). Furthermore, we do not want to assume the data to be normally distributed, because it is not clear why it should be. The Wilcoxon test fulfills all those properties and can handle the rather small data set of 15 training runs. The user study was performed by 5 participants. All of them were male and between 21 and 29 years old. Three have

completed their Bachelor, one has a Completed Vocational Training and one has a University Entrance Qualification. Only one of them knew what Reinforcement Learning is, and two had heard the term before but didn't know what it means. Each participant performed one familiarization phase, followed by three training runs, for each algorithm. One training consists of 7 episodes. The order in which the algorithms were used was chosen randomly before each experiment.

The familiarization phase consists of one episode that is only ended when the car reaches the goal or the user quits the training. Here the participant shall understand the principal goal of the task and how the training works. During this phase, he can ask questions to understand this. On the other hand, the episode is not enough for the participant to optimize its behavior.

During the actual experiments, the episodes end after 200 time steps. The car is reset and a short message is displayed. To enable human interaction, there is a thumb up and a thumb down button when feedback shall be given, and one button for each possible action when advice shall be given. The acceleration actions are visualized by an arrow in the corresponding direction and the no acceleration action by a dot.

To give feedback, $\min_{\text{react}} = 0.2\text{s}$ as minimal and $\max_{\text{react}} = 0.9\text{s}$ as maximal reaction time was assumed. For both algorithms a learning rate of 0.05 and the same RBF features as in the simulation experiments were used.

A visualization of the results can be seen in Figure 5.9. They were created similarly as for the simulation results, described in 5.2. Only for the confidence interval around the learning curves, the necessary confidence was reduced from 95% to 80% to make the plots easier readable. It is natural that the confidence decreases in the user studies compared to the simulations because the humans introduced additional noise and fewer training runs were available. The SARSA baseline was not included in these plots, because the car did not reach the goal at all in the first seven episodes, thus the learning curve is constantly -200 in this area. To visualize the number of interactions that were performed in the study, the averages over all runs were plotted for each episode. The confidence interval was left away completely to make it better readable.

It can be seen that the task can be learned better with the advice algorithm in this setting.

The average cumulative reward for the 7 training episodes was for the TAMER -1366.1 and -1117.9 for the Advice algorithm. A Wilcoxon test leads to a p value of 0.00081, which is clearly smaller than the chosen level of significance of 0.01. Thus, the null hypothesis can be refused and Advice performs better during the training phase with a level of significance of 0.01.

The impression that the Advice algorithm performed better is also shared by the participants: The question after both experiments *Which method led to better results, according to your impression* was answered by all five participants with *By advising actions*. Besides, most of them found it generally easier to provide advice. The question *What kind of information was easier for you to provide, independent from the behavior of the car* was answered by four of five participants with *It was easier to advise actions*. Only one answered *Both were equally easy (difficult)*.

But the performance of the policy learned with Advice is not only better during the training phase. The learned models were evaluated afterwards. Here an episode is considered to be successful if the car reached the goal position in less than 200 steps. Each learned model was evaluated for 100 episodes. In those evaluations, a success rate of 18.4% for the models trained with the TAMER algorithm and 90.2% for the ones resulting from Advice were achieved. This shows that it is possible to train the desired behavior with the Advice algorithm, but not with the TAMER algorithm in this setting.

It appears in Figure 5.9 that the number of interactions is more or less the same for both algorithms. This can be proven with the Wilcoxon test on the number of interactions that leads to a p value of 0.97734. Thus, the null hypothesis that both algorithms lead to the same amount of interactions can not be rejected.

One training run took with TAMER 128s and with Advice 99s. The Advice algorithm was faster because it led to more successful episodes.

The amount of given feedback was roughly constant during the training and there was more negative than positive feedback. This makes sense because most episodes with TAMER were not successful. In contrast to that, the Advice algorithm was already regularly successful during the training. However, the amount of given advice was reduced rather slightly. Watching the experiments it could be noted, that most participants just kept on giving advice, even when the behavior was already learned. Besides this, it can be seen that the no acceleration button is never pressed. This makes sense because it is not useful to perform no acceleration.

One problem, that could regularly be seen during the experiments with TAMER, was that the participants got stuck with one direction. Also, one participant commented in the *what was your impression* text box: „Eine Richtung wurde nicht mehr angezeigt“ (One direction was not displayed anymore), referring to this problem. This indicates that the algorithm is not robust enough against wrong feedback. This problem could not be observed in the simulated experiments, because the oracles did not change their strategy to give feedback during the training.

5.3.2 Mountain Car (2)

Because in the first user study appeared the problem that the agent got stuck with a single action we changed the feature representation of state action pairs to make the learning process more robust against bad information. Besides, the

possibility to reset an episode or to finish the whole training run was added. One result of the first user study was that the participants did not really stop to interact. This way it could not be seen, whether they thought that the problem was solved. This is why we added the possibility to completely stop the training. All in all, we want to answer the following questions with the second user study:

- **Can the behavior** - that the car drives up to the flag - **be taught** using the TAMER and the Advice algorithm?
Does the car reach the flag within 200 time steps following the learned model without interaction with a human in at least 80% of the trials?
- Does one of the two ways to train the agent rather lead the participants to **actively finish** the training or **reduce** the amount of **interaction**?
Perform a Wilcoxon test, whether human interaction was reduced during the training.

The user study was again performed by 5 participants with the following properties. Four of them were male and one female. Four were between 21 and 29 years old, while one was between 40 and 49. Two have completed their Bachelor, two have a Completed Vocational Training and one has a University Entrance Qualification. Only one of them had heard the term Reinforcement Learning before but didn't know what it means.

The procedure was similar to the first user study. One difference was that the participants could finish the training early when they had the feeling that the agent has learned everything. This can lead to a shorter training phase. Besides, we wanted to give the participants a bit more time between the episodes. For that reason, the message that the agent is currently reset to a new start position is displayed for a longer time (1s to now 5s). To make sure that the participants are ready when the training continues, a countdown is displayed, which shows the time until the next episode.

A visualization of the results of this user study can be seen in Figure 5.10. When a training run was finished early by the user, there is no data about the gained reward for the avoided episodes available. To be still able to plot the learning curves, those values were filled up by the result of the evaluation of the learned model of this specific try. This leads to a similar result as if the participant would have simply watched the last episodes without interacting. The only difference is that the mean of 100 evaluations instead of one random evaluation is used.

The learned models led to an average success rate of 75% with the Advice algorithm and 72% with the TAMER algorithm. Thus, the goal of 80% is reached by neither of the algorithms. Nevertheless, it is a drastic improvement for the TAMER algorithm, which achieved a success rate of only 18.4% in the first user study. On the other hand, the evaluation of the models trained with Advice is worse than in the first user study. Considering that the learning curves for the Advice algorithm look quite similar in both user studies, one possible explanation for the different performances in the evaluation would be that the model based on the features of the second study was not able to capture the taught policy as well as the model based on the features of the first study. During the training, the participants could influence the behavior of the agent by their advice such that there is no big difference in the learning curves. But in the evaluation, only the learned model is used and if some, maybe even small, aspect of the policy is not captured well there, it can lead to worse performance.

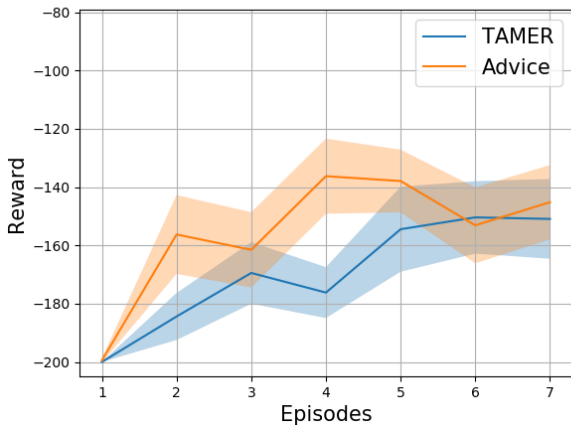
With the TAMER algorithm, 5 training runs were actively finished by users, while the Advice algorithm only led to 2 actively finished training runs. In Figure 5.10 can also be seen that the amount of given feedback (good, bad) decreased more than the amount of given advice (left, no, right). In contrast to the first study, the *no acceleration* button is used this time. However, it is still used only seldom.

To show whether the interaction is really reduced, we perform a Wilcoxon test on the following hypothesis. For each training run, we add up the amount of all human interaction of the first two and the last two episodes. The null hypothesis is that the amount of interaction is the same at the beginning (the first two episodes) and at the end (the last two episodes) of the training. For the Advice algorithm, we get a p value of 0.0883 on this test, which means that we can not falsify the null hypothesis when we keep the level of significance of 0.01. On the other hand, for the training runs with TAMER, the p value is 0.0059, and we can falsify the null hypothesis that the human interaction stays the same during the training with a level of significance of 0.01. So, we can show that the participants reduced their interaction during a training run when giving feedback, but we can not show this for their interaction with Advice.

It seems that humans rather tend to reduce their interaction or even stop the training when they give feedback. A reason for this could be that it is necessary to take care of the current behavior to be able to give feedback for it. In contrast to this, advice can principally also be given independently from the used policy. Although this is not how it's meant to be, some participants may have concentrated less on the agent's behavior when they gave advice, and thus did not reduce their interactions when the policy improved.

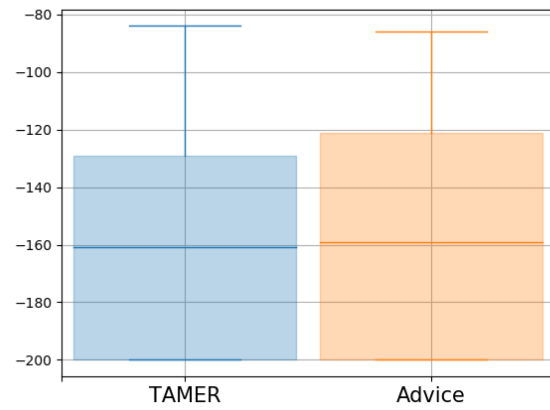
One training run took with TAMER 139s and with Advice 116s in the second user study. This is longer compared to the first one because the time that is waited after each episode is longer. The Advice algorithm still took less training time than the TAMER algorithm, although training with TAMER was more often finished early by the users.

In contrast to the first user study, the impressions of the participants were much more mixed here. Two participants preferred giving feedback and two preferred giving advice. One didn't have priorities for one of those. All participants



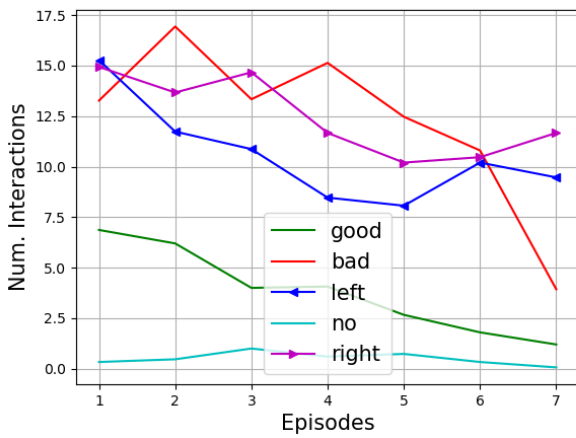
(a) Learning Curve

Mean episodic reward with 80% confidence interval



(b) Final Behavior of the Trained Models

Boxplots with median and quartiles of evaluation results of the learned model



(c) Average Number of Interactions

Here the amount of human feedback (good, bad) and advice (left, no, right) that was given during the user study can be seen.

Here the results of the second user study that used the Mountain Car environment can be seen. Both algorithms performed similarly well. Advice only leads to slightly better returns at the begin of the training. The amount of given feedback is reduced during the training this time.

Figure 5.10.: Results of the Second User Study

answered both questions, which method led to better results and which method was easier to use, the same. I.e. if one found it easier to give feedback, he also thought that the corresponding results were better and vice versa. There were also explanations given, why one way of interaction was easier to use. One participant wrote that giving feedback was easier for him because the acceleration was visualized here („fiel mir persönlich leichter, da die Beschleunigung visualisiert wurde.“). On the other hand, another participant wrote that the visualization of the accelerations made the interaction more difficult because two different things have to be watched here, the car itself and the visualization of the acceleration („Beim Feedback musste man zwei Anzeigen beobachten: Das Auto und die Anzeige über den Motor. Anhand dieser Beiden sich für ein Feedback entscheiden. Beim Aktionen vorschlagen musste man nur aufs Auto achten. Das war leichter.“).

The way the participants interact can differ quite much. In Figure 5.11 are the strategies of two participants of this user study depicted as an example.

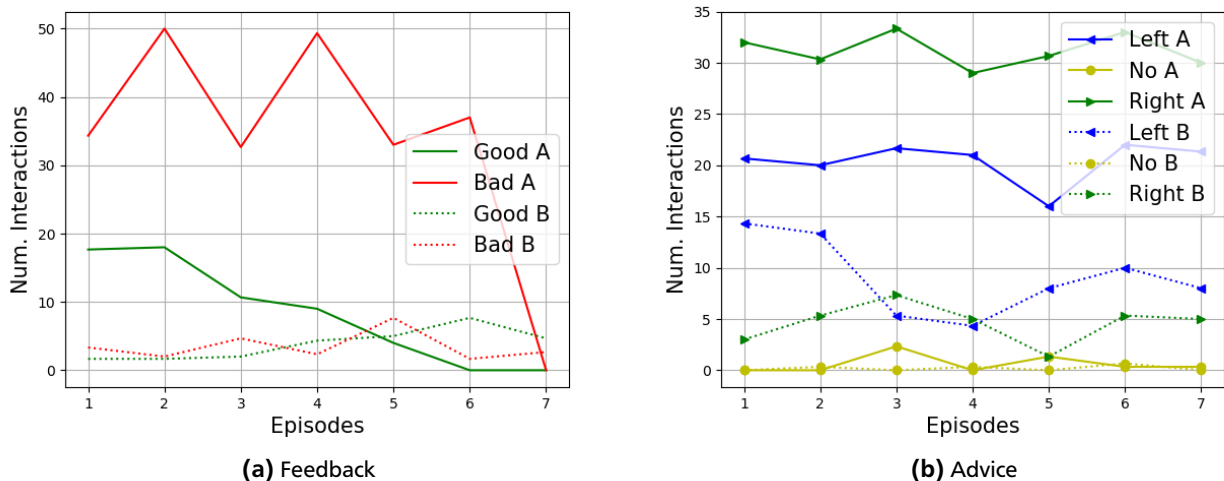


Figure 5.11.: Comparison of Interaction Strategies

The amount of interactions is visualized for two participants, denoted A and B. The curves were averaged over all training runs for each participant.

It can be seen, that participant A tends to interact much more than participant B. This is a challenge for Interactive Reinforcement Learning algorithms because they have to be able to handle both well. For example, a small learning rate might be advantageous for participant A, because it avoids that the interactions of the last episodes completely overwrites the things that were learned in the episodes before. On the other hand, participant B would probably not like this small learning rate, because the agent would not adapt its behavior as much to the few participant’s interactions as it would with a higher learning rate. Thus, it is important to find algorithms and hyperparameters that can handle as many different user behaviors as possible.

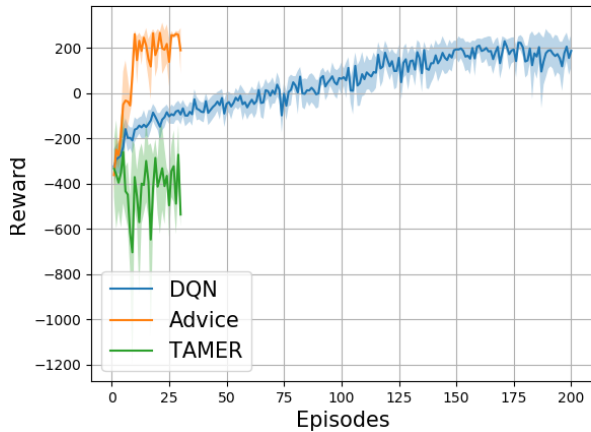
5.3.3 Lunar Lander - Expert Trainer

The Lunar Lander is a more complex environment than the Mountain Car and thus is not that easy to understand by a non-expert trainer. Because of this, it is interesting to compare the results of an expert with the ones of non-experts. I performed the expert study myself. To become an expert, I first got familiar with the environment by many training trials with different algorithms and settings. Afterwards, I performed some runs and saved their results. Those are presented in this section.

For the Lunar Lander environment, the deep versions of the algorithms are used. All of them have a neural network with two hidden layers of 150 and 120 nodes. Because the network can capture more complex models than a linear model, no additional feature extraction was used here.

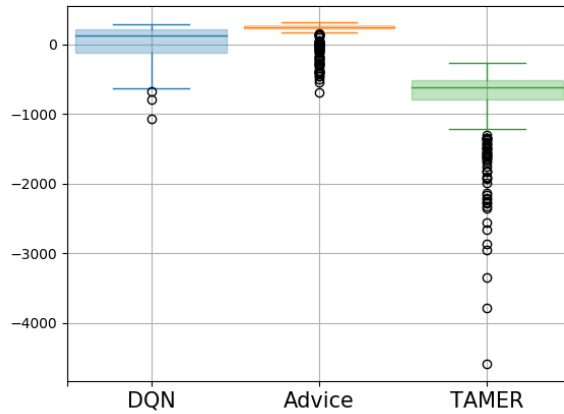
For the DQN baseline, 20 training runs with 200 episodes were performed, while for the interactive algorithms only 5 training runs were performed due to time reasons. When only the human interaction and no environmental reward were used, the training took 30 episodes, when human interaction was combined with the environmental reward, we used 50 episodes. A learning rate of $\alpha = 0.001$ and a batch size of $N = 64$ were used for all algorithms. In DQN the target network was updated every 25th step and $\gamma = 0.99$ was used. Human feedback is assumed to refer to all actions between the minimal $\min_{\text{react}} = 0.2s$ and the maximal reaction time $\max_{\text{react}} = 0.9s$. For DQN an ϵ -greedy policy with a linearly

decreasing ϵ from 0.5 to 0 was used. When human interaction is used, only the DeepTAMER without a combination with environmental reward needed randomness from an ϵ -greedy policy with a linearly decreased ϵ (0.2 to 0). All other algorithms just used a greedy policy.



(a) Learning Curve

Mean episodic reward with 80% confidence interval



(b) Final Behavior of the Trained Models

Boxplots with median and quartiles of evaluation results of the learned model

Figure 5.12.: Results of expert training on Lunar Lander without environmental reward

The model trained with Advice can outperform the DQN baseline already after a much shorter training time. On the other hand, it was not possible to train a good policy with TAMER.

When learning only from human interactions (Figure 5.12), a good behavior - an episodic reward of about 200 - is already reached after 10 episodes with Advice. On the other hand, it was not possible to train the behavior only with human feedback. The achieved behavior during training was here even worse than random. The evaluation of the learned model shows a similar result. The Advice model clearly outperforms the model trained with DQN, while the model trained with TAMER performs much worse than the DQN one.

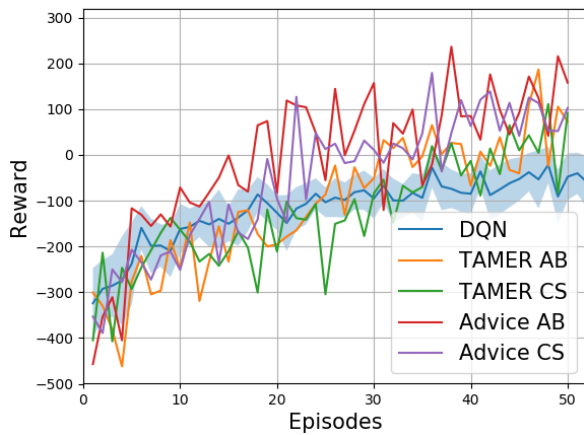
When learning only from human feedback, it can happen that some action is not executed anymore. This could already be observed similarly in the first user study with the Mountain Car environment, but it is even worse here, also because there are four instead of two relevant actions in this environment. We introduced an initial randomness of 20% that is linearly reduced during the training, to overcome this problem, but it could not completely be solved this way.

The learning behavior of the algorithms that use environmental reward as well as human interaction is naturally more similar to the one of the Reinforcement Learning baseline. A visualization can be seen in Figure 5.13.

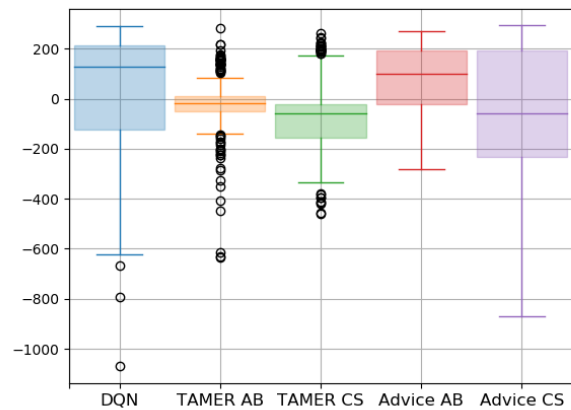
We tried out both methods to combine human interaction with environmental reward, Action Biasing (AB) and Control Sharing (CS) for the Lunar Lander environment. The fast switching between actions, that is caused by the use of Control Sharing and that was adversely for the Mountain Car environment, is no problem for the Lunar Lander, because the actions have to be switched with a relatively high frequency anyhow. We used the following settings for the fusion of the models, all other hyperparameters were kept as in the experiment only with human interaction. For both experiments, an accumulation factor of $A = 0.7$ and a diminishing factor of $\lambda = 0.9$ were used. The weight for models learned from human information differed between the fusion methods. For Action Biasing we used $C = 50$ and for Control Sharing $C = 2$.

Algorithms that combine multiple kinds of information learn multiple models in our framework. In this concrete user study, the Simultaneous Advice algorithm learns one model of environmental reward Q and one model of given human advice A . Analogically a model of environmental reward Q and a model of human feedback F is learned by the Simultaneous TAMER algorithm. During the training, the combination of the models is clearly defined. If the human trainer would stop to interact, the Q model would completely overtake the action selection after some time. Because of this, we used only the Q for the evaluation of all algorithms that combine human interaction with environmental reward.

All methods that combined expert interaction with environmental reward improved the behavior at the end of the human training compared with the same episodes of DQN learning. On the other hand, the learned models could not perform clearly better than the DQN baseline in the evaluation. But it can be seen that they rather avoid episodes with very bad rewards, even though the DQN model was trained for four times as many episodes. The Simultaneous Advice algorithm seems to perform best out of the interactive algorithms that used the environmental reward. During the learning, it is the



(a) Learning Curve



(b) Final Behavior of the Trained Models

Mean episodic reward with 80% confidence interval. Only Boxplots with median and quartiles of evaluation results of the begin of the DQN learning curve is shown. For the complete one see Figure 5.12.

Figure 5.13.: Results of expert training on Lunar Lander with environmental reward

The human interaction can improve the agent’s behavior during the training. However, in the evaluation most learned models could not keep up with the DQN baseline.

one that first persistently rises above the DQN baseline and in the evaluation, it is the only one whose median is nearly as good as the DQN baseline.

	DQN	TAMER	Advice	TAMER AB	TAMER CS	Advice AB	Advice CS
Mean Training Reward	-120	-430	154	-131	-119	-7	-73
Mean Evaluation Reward	44	-771	193	-23	-73	80	-94
Evaluation Success Rate (%)	63.6	0	83.6	7.2	12	48.4	41.2

Table 5.3.: Results of Expert Training on Lunar Lander

Three basic types of information are presented: The Mean Training reward is the episodic reward of the training phase averaged over all training episodes and training runs. For DQN only the first 50 episodes were used because the other algorithms were trained for at most 50 episodes. The Mean Evaluation Reward is the episodic reward over training runs and evaluation episodes. The Evaluation Success Rate is also averaged over training runs. An episode is assumed to be successful if the lander comes to rest without a crash.

A summary of all results of the expert training on the Lunar Lander is presented in Table 5.3. There are three performance categories listed. First the average episodic reward during the training, which shall show whether human information could improve the behavior during the training. Second the average episodic reward in the evaluation of the completely trained models, which shall show whether the learned model can really be improved by the human interaction during the training. The third category shall show another aspect of the learned models. The main goal for the human trainer was to get the lander safely to the moon. With the ratio of episodes that succeeded in reaching this goal in the evaluation, we want to show how well this specific aspect was learned.

The Advice algorithm achieves the best results in all three listed categories, while TAMER is worst, also in all three categories. Both Simultaneous TAMER versions performed similarly. During the training they the overall reward does not differ much from the DQN baseline, and the learned models are even worse than the baseline, regarding the reward as well as the success rate. The two Simultaneous Advice versions can both improve the DQN behavior during the training, the Action Biasing one even more than the Control Sharing one. In the evaluation, Simultaneous Advice with Action Biasing can improve the mean episodic reward a bit while it worsens it with Control Sharing. The success rates are similar for both versions but worse than the baseline. It is interesting to note that Simultaneous Advice with Action Biasing could improve the average reward in the evaluation, but worsened the success rate in the same episodes, although the main goal of the trainer was, to achieve a high success rate. One explanation would be that the human

trainer focuses more on getting the lander towards the goal position because he wants the agent to succeed as early as possible. In contrast to that, a stable flight seems to be more important for the DQN algorithm, which can lead to a higher success rate in the long term. One aspect of this problem might be that a human has only limited patience. The trainer does not want to watch the agent to flight in the same position for a very long time.

5.3.4 Lunar Lander - Non-Expert Trainer

After the expert experiments, we performed a user study with non-experts on the Lunar Lander environment. There were two participants, both between 50 and 59 years old and both with a completed vocational training. One of them was male, the other one female. Both have only heard of Reinforcement Learning but did not know what it is.

We decided to use the Simultaneous Advice and Simultaneous TAMER algorithms, both with Action Biasing in this user study. On the one hand, the Advice algorithm without environmental reward performed best in the expert trainer experiments. So, one might think that this algorithm would also be best for the non-expert user study. But on the other hand, this performance was only possible, because the trainer knew beforehand, what a good policy looks like. A non-expert user first has to get to know the dynamics of the lander. For this purpose, the environmental reward can support the human. We kept the settings of the algorithms the same as for the expert user study.

Each participant performed only one run with each algorithm because this took already about half an hour per participant. Additional time was necessary to read and understand the instructions and fill out the questions of the form.

	DQN	TAMER AB (1)	Advice AB (1)	TAMER AB (2)	Advice AB (2)
Mean Training Reward	-120	-175 / -143	-119 / -247	-156 / -149	-205 / -16
Mean Evaluation Reward	44	-147 / -500	-163 / -143	-313 / -311	-387 / -203
Evaluation Success Rate (%)	63.6	0 / 0	1 / 0	0 / 1	0 / 44

Table 5.4.: Results of Non-Expert Training on Lunar Lander

Each column contains two values for the two human trainers. The number in parentheses after the algorithm name denotes, whether the first or the second training run is meant.

Three basic types of information are presented: The Mean Training reward is the episodic reward of the training phase averaged over all training episodes. For DQN only the first 50 episodes were used because the other algorithms were trained for 50 episodes. The Mean Evaluation Reward is the episodic reward averaged over evaluation episodes. The Evaluation Success Rate assumes an episode to be successful if the lander comes to rest without a crash.

Both participants were not able to teach the behavior. Only for one learned model, a success rate of 1% could be achieved, and all other models did not lead to a single successful episode in the evaluation. However one of the participants noted that he had gained a better understanding of the behavior of the agent during the experiment: „Mit zunehmender Dauer bekommt man ein besseres Gefühl für das Flugverhalten des Ufos“.

To see whether the participants have really learned something during this first experiment, both of them repeated the exact same user study one week later. One of them could improve the performance of the Advice algorithm, compared to the first try and achieved a quite good success rate of 44% in evaluation. Besides, the mean episodic reward during the training could be improved in comparison with the DQN baseline.

Table 5.4 gives an overview of the results of the non-expert user study. We did not include plots here, because there was only relatively little data that would have been hard to read in the plots.

5.3.5 Taxi like Gridworld with Holes

The last user study we performed was done on the Taxi like Gridworld. In contrast to the Mountain Car and the Lunar Lander environment, the Taxi like Gridworld with holes has only discrete states and the single actions can be judged distinctly. Another important difference is that we visualize the action that the agent currently plans to perform to the human trainer. To do this, an arrow points from the taxi in the direction the agent wants to make the next step to.

For this user study, two algorithms are used. With TAMER the action that is currently proposed, i.e. the direction that the arrow currently points to, shall be judged by the human. The other used algorithm is Supervised Advice. This means that advice is immediately followed if it is given at the current time step, else the learned advice model is followed.

In a real-world application, a human trainer would not want to interact with the agent the whole time. Thus, we limited the number of interactions in this study to see whether the users are able to train the task already with little interaction. For Supervised Advice as well as for TAMER only 20 interactions per episode were possible. The amount of interaction the user has left in the current episode was displayed in the user interface.

For the Lunar Lander and the Mountain Car, the time waited between two actions are performed was chosen to make the movement look fluent. In contrast to those environments, there is no continuing movement in a gridworld. Each action is distinct from all others. Thus, a good time that on the one hand enables the human to interact and on the other hand does not bore the human, has to be chosen. We try out two different durations to wait between two actions in this user study to evaluate what might be well suited. In the slower version, that gives the human more time to react, 1.2 seconds pass between two actions. In the faster version, there are only 0.8 seconds between two actions. We also want to see, whether there are differences in the required time between the algorithms.

In the user study, the slower version is used first. Afterwards, the speed is increased.

The user study was again performed by 5 participants. All of them were between 21 and 29 years old. Four of them were male and one female. One has completed his Bachelor, two have a Completed Vocational Training and two have a University Entrance Qualification. Only two of them had heard the term Reinforcement Learning before but didn't know what it means. Those two had participated in the first Mountain Car user study already.

Similarly to the second Mountain Car user study, the participants had the possibility to finish the training, when they had the impression that the agent has learned everything, and they could reset an episode. In the second Mountain Car user study, there was never an episode reset because there was no reason to reset an episode. However, in this user study with the gridworld, it makes sense to reset the episode when there are no interactions left for the current episode.

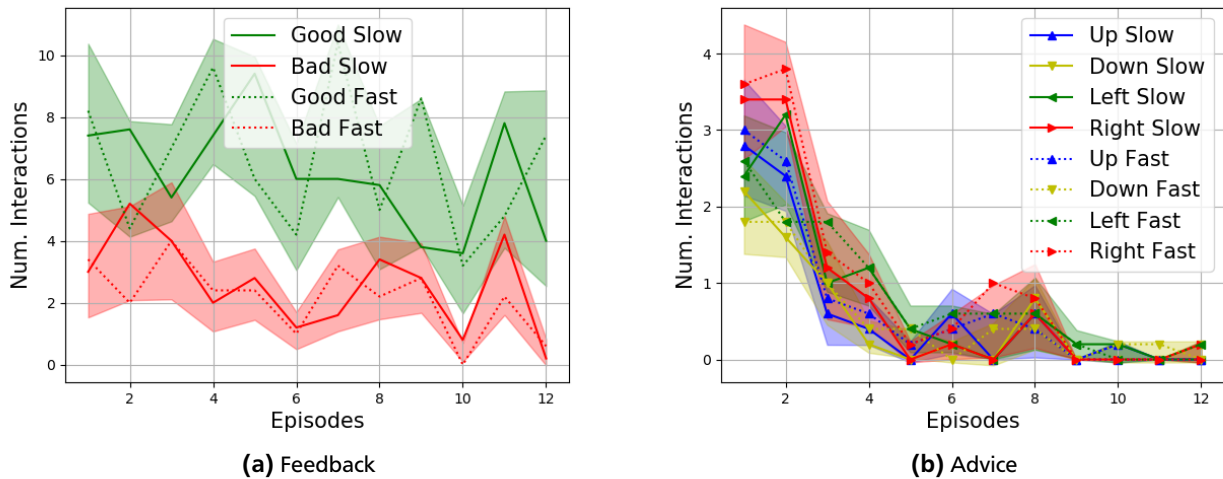


Figure 5.14.: Number of Interactions on the Taxi Like Gridworld

The average number of interactions per episode during training in the taxi like gridworld environment. There is a distinct line for both, the fast and the slow version of the experiment. The plotted shadow shows the 80% confidence interval over the data of both versions together.

When an episode is reset or the training is completely finished, there is no data available that would be needed for plotting a learning curve. Because quite many episodes were reset and some training runs were finished quite early we decided to not plot a learning curve for this user study. Nevertheless, there are other measures that can describe the learning process:

When the TAMER algorithm was used, the agent could only seldom bring all passengers to the goal. In the 12 trials, it was achieved only 1.8 times in the slow version and 1.2 times in the fast version on average. Because it was so hard to reach the goal, the training was never finished early by the participants with the TAMER algorithm in the slow version and nearly never in the fast version, where a training run lasted on average for 11.6 episodes. With the Supervised Advice algorithm, the training was much more successful. Thus, the participants rather tended to finish the training early. In the slow version, a training took on average only 7.6 episodes, out of whom 7.2 were successful. The fast version seems to be a bit harder to train with supervised Advice. Here a training run lasted for 9.2 episodes with 8 successes on average. So, the participants wanted to train the agent for more episodes and had a slightly worse success rate here.

In Figure 5.14 can be seen that the amount of interactions and its development during the training does not differ much between the slow and the fast version. Besides, it can be seen that the amount of advice is reduced more during the training than the amount of given feedback. This makes sense because the agents that were trained with advice performed much better in the training than those who were trained with feedback.

In the evaluation, it can be seen that the models trained with Supervised Advice again outperformed the ones trained with TAMER. With TAMER the experiment with a slower action frequency led to better results as it was already observed in the

performance during the training. Interestingly, the Supervised Advice algorithm could achieve better evaluation results when it was used with a higher action frequency. This combination of Supervised Advice with a fast action frequency was the only one that could completely avoid running into holes in the evaluation. See Figure 5.15 for a visualization of the evaluation.

The answers of the participants in the user study support the findings above. All of them wrote that it was easier to provide advice than to give feedback. Four of five also had the impression that giving advice leads to better results and for the other one the performances of both methods appeared to be the same. Four participants of the user study found the slower version of the experiment more pleasant to give feedback. Only one preferred the fast one. For giving advice, the answers are more mixed. Only for two users, the slower version was more pleasant and for two users both were similar. Again, one preferred the fast version.

One training run with the giving feedback and the slower action frequency took on average 8.6 minutes. This can be reduced to 6 minutes with the higher action frequency. When advice was used as human interaction, a training run took on average 6.1 minutes with the slower action frequency and 4.3 minutes with the faster version.

All in all, the slower action frequency seems to be better suited for giving feedback. For giving advice this is not so clear. On the one hand, the performance during the training was slightly better with the slower version. On the other hand, the faster action frequency led to better results in the evaluation. One advantage of the faster version is of course that it can reduce the overall training time. This is an important factor that must not be neglected because the time the human has to interact with the system is typically costly.

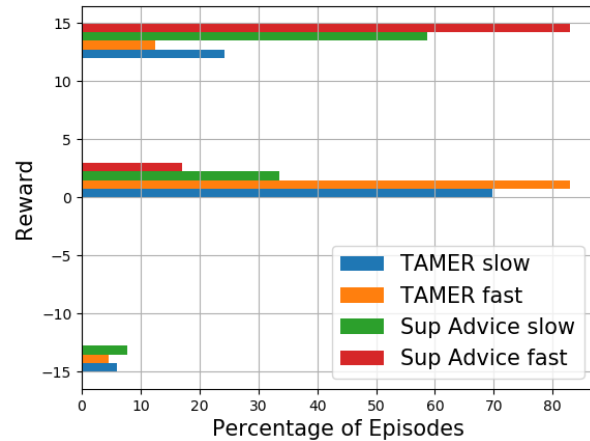


Figure 5.15.: Evaluation of Learned Models for the Taxi like Gridworld Environment

The models were evaluated for 100 episodes each. There occur only three different episodic returns, and therefore we used a histogram to visualize them. A reward of 15 is reached when the task was solved. When the agent got stuck somewhere in the grid or got to the goal without passengers, this is represented by a reward of 0 in the histogram. A reward of -15 is caused when the agent runs into a fire.

6 Conclusion and Future Work

When Reinforcement Learning shall help to bring robots in our everyday life, there are still multiple challenging problems. One of them is, that it is hard, especially for non-expert users, to define good reward functions. But if the tasks have to be adopted, for example to the special properties of a household where the robot shall operate or to the needs of the user, this might require an adapted reward function each time. Besides, the needs of the users might even change over time. Because in such scenarios there are humans in the environment of the robot anyhow, a natural idea is to let the human interact with the robot to improve the learning, or to learn from the human interaction instead of a predefined reward function. This is then frequently called Interactive Reinforcement Learning.

There is already literature that utilizes different approaches in the field of Interactive Reinforcement Learning. In this thesis a unified framework that can combine algorithms for learning from human feedback, human advice, and an environmental reward function. The main requirement is only that some kind of a value model is learned, which is the case for almost all Interactive Reinforcement Learning algorithms.

Some algorithm combinations were implemented and a user interface was designed.

In the experimental part, the algorithms were first evaluated with the use of oracles that simulated the human interaction. Here the interactive algorithms clearly outperformed the Reinforcement Learning baseline on the gridworld environments. It was possible to train the desired behavior without an environmental reward function. On the Mountain Car environment, Interactive Reinforcement Learning algorithms were used with and without the reward function. They could speed up the learning process, but when the resulting models were evaluated, the results of Interactive Reinforcement Learning could not fully reach the ones of the Reinforcement Learning baseline.

In those experiments without real humans, the use of simulated feedback and advice performed very similarly. However, experiments where the human interaction is only simulated can never show how the algorithms perform with real humans. Because of this, user studies with real humans were performed as a next step. With them, we wanted to answer the research questions of the thesis.

6.1 Research Questions

At the begin of the thesis, three research questions were asked, that we wanted to answer with this work. We want to refer to them again to show what we could answer:

1. Can human feedback enable or improve the training of agents?

In the user study on the taxi like gridworld, the participants could train the agent to bring all passengers to the goal. Giving advice worked much better than giving feedback, but also feedback improved the Reinforcement Learning baseline in this regard. In the second Mountain Car user studies, the training could be improved with advice as well as feedback, in the sense that a success rate of about three quarters could be reached in the evaluation after only seven training episodes. the Reinforcement Learning baseline never reaches the goal after such a short training. In the first Mountain Car user study, even the beforehand fixed goal performance of 80% could be reached with giving advice. On the more complex Lunar Lander environment, it was much harder for humans to improve the learning. The expert trainer was able to slightly speed it up with advice as well as with feedback when it was combined with the reward function. Without the reward function giving advice worked very well and clearly outperformed the Reinforcement Learning baseline, but giving feedback only worsened results. For the non-experts, it was very hard to improve on the baseline.

2. How can different ways of interaction and different interpretations of human feedback be beneficial for the training?

In our experiments giving advice basically always worked best. However, it is important to memorize the given advice and not only perform it once. This was already shown in the experiments without real humans, where the Advice algorithm performed better than Supervised Reinforcement Learning.

3. Is there a clear relationship between different problem settings and corresponding best working methods?

We could not find a clear relation here, because giving advice worked always best. One finding is that it can be hard to get the correct action performed when giving advice because, with bad luck, all other actions have to be given negative feedback before. This problem naturally increases with the size of the action space. On the other hand, it becomes harder to design a useful interaction interface for giving advice when the action space becomes very big. So there is also no clear relation here.

6.2 Future Work

There are different ways that could be followed after this thesis. One possibility would be to perform experiments with other algorithms. Algorithms that interpret binary human feedback as correctness of performed actions could be directly used with the presented framework. Besides, for example, preference based algorithms could also be included. Adding more algorithms is important to show how generally the framework can really be used. Moreover, there is the chance that other algorithms perform even better and therefore more complex tasks could be solved.

A, probably more time consuming, alternative would be to try out the framework on real physical systems. There is a difference between judging some behavior that is displayed on a screen and a robot that interacts in the real world. The use of Interactive Reinforcement Learning was heavily motivated by robot learning tasks, thus it would be important to see how well interaction can be performed by non-expert users in such tasks to justify that Interactive Reinforcement Learning is really useful.

Besides, there are environment types that were not captured in this thesis. For example environments with continuous actions would be an interesting challenge, because most existing Interactive Reinforcement Learning algorithms, as well as the framework proposed in this thesis, assume only discrete actions. Referring to the previous point of bringing Interactive Reinforcement Learning to real robots, this becomes very important. High-level control, to decide which task to perform next, can be represented well with a discrete and finite action space. Though, when the robot shall be completely controlled by the learned model this would typically require a multidimensional continuous action space. A well designed discretization would be a simple approach to this problem. If this is not sufficient, a more sophisticated optimization over the action space would be necessary to select actions in such scenarios.

Another challenge that occurs with continuous actions is to find a way how actions can practically be advised. For real robot systems, kinesthetic teaching might be a possibility. An alternative could be, to specify to which coarse direction the end effector or some joint should move. This could, for example, be implemented with a natural language interface, which could potentially be more convenient for a user than physically moving a robot arm.

In the user studies of this thesis, it could be seen that the participants used different strategies and had different preferences regarding the way of interaction. For example, some users provided much more interaction than others. Besides, in the second user study with the Mountain Car environment, some participants argued that it was easier to provide advice, while others could explain, why it was easier for them to give feedback. Therefore, another direction of further research would be to find more ways to better incorporate personal preferences of the specific user. For example, one could let the user decide which kind of interaction he wants to use.

Bibliography

- [1] D. Abel, J. Salvatier, A. Stuhlmüller, and O. Evans, “Agent-agnostic human-in-the-loop reinforcement learning,” *arXiv preprint arXiv:1701.04079*, 2017.
- [2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
- [3] 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,” *arXiv preprint arXiv:1509.02971*, 2015.
- [4] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International Conference on Machine Learning*, pp. 1889–1897, 2015.
- [5] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.
- [6] S. J. Russell, “Learning agents for uncertain environments,” in *COLT*, vol. 98, pp. 101–103, 1998.
- [7] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in *Aaai*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.
- [8] A. Boularias, J. Kober, and J. Peters, “Relative entropy inverse reinforcement learning,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 182–189, 2011.
- [9] S. Schaal, “Learning from demonstration,” in *Advances in neural information processing systems*, pp. 1040–1046, 1997.
- [10] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, “Keyframe-based learning from demonstration,” *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.
- [11] A. L. Thomaz, G. Hoffman, and C. Breazeal, “Real-time interactive reinforcement learning for robots,” in *AAAI 2005 workshop on human comprehensible machine learning*, 2005.
- [12] W. B. Knox and P. Stone, “Interactively shaping agents via human reinforcement: The tamer framework,” in *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16, ACM, 2009.
- [13] D. Arumugam, J. K. Lee, S. Saskin, and M. L. Littman, “Deep reinforcement learning from policy-dependent human feedback,” *arXiv preprint arXiv:1902.04257*, 2019.
- [14] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, “Policy shaping: Integrating human feedback with reinforcement learning,” in *Advances in neural information processing systems*, pp. 2625–2633, 2013.
- [15] R. Loftin, B. Peng, J. MacGlashan, M. L. Littman, M. E. Taylor, J. Huang, and D. L. Roberts, “Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning,” *Autonomous agents and multi-agent systems*, vol. 30, no. 1, pp. 30–59, 2016.
- [16] B. Peng, J. MacGlashan, R. Loftin, M. L. Littman, D. L. Roberts, and M. E. Taylor, “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*, pp. 957–965, International Foundation for Autonomous Agents and Multiagent Systems, 2016.
- [17] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, “End-to-end robotic reinforcement learning without reward engineering,” *arXiv preprint arXiv:1904.07854*, 2019.
- [18] J. MacGlashan, M. K. Ho, R. Loftin, B. Peng, G. Wang, D. L. Roberts, M. E. Taylor, and M. L. Littman, “Interactive learning from policy-dependent human feedback,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2285–2294, JMLR. org, 2017.

-
- [19] G. Warnell, N. Waytowich, V. Lawhern, and P. Stone, “Deep tamer: Interactive agent shaping in high-dimensional state spaces,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [20] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters, “Active reward learning,” in *Robotics: Science and systems*, 2014.
- [21] A. Kupcsik, D. Hsu, and W. S. Lee, “Learning dynamic robot-to-human object handover from human feedback,” *arXiv preprint arXiv:1603.06390*, 2016.
- [22] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, “Learning trajectory preferences for manipulators via iterative improvement,” in *Advances in neural information processing systems*, pp. 575–583, 2013.
- [23] R. Pinsler, R. Akrou, T. Osa, J. Peters, and G. Neumann, “Sample and feedback efficient hierarchical reinforcement learning from human preferences,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 596–601, IEEE, 2018.
- [24] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” in *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017.
- [25] C. Wirth, J. Fürnkranz, and G. Neumann, “Model-free preference-based reinforcement learning,” 2016.
- [26] A. Bajcsy, D. P. Losey, M. K. O’Malley, and A. D. Dragan, “Learning from physical human corrections, one feature at a time,” in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 141–149, ACM, 2018.
- [27] A. Qureshi, *DEEP REINFORCEMENT LEARNING FOR HUMAN-ROBOT INTERACTION IN THE REAL-WORLD*. PhD thesis, 09 2017.
- [28] S. K. Kim, E. A. Kirchner, A. Stefes, and F. Kirchner, “Intrinsic interactive reinforcement learning—using error-related potentials for real world human-robot interaction,” *Scientific reports*, vol. 7, no. 1, p. 17562, 2017.
- [29] H. Ritschel, T. Baur, and E. André, “Adapting a robot’s linguistic style based on socially-aware reinforcement learning,” in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 378–384, IEEE, 2017.
- [30] M. Khamassi, G. Velentzas, T. Tsitsimis, and C. Tzafestas, “Robot fast adaptation to changes in human engagement during simulated dynamic social interaction with active exploration in parameterized reinforcement learning,” *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 4, pp. 881–893, 2018.
- [31] A. L. Thomaz and C. Breazeal, “Teachable robots: Understanding human teaching behavior to build more effective robot learners,” *Artificial Intelligence*, vol. 172, no. 6-7, pp. 716–737, 2008.
- [32] H. B. Suay and S. Chernova, “Effect of human guidance and state space size on interactive reinforcement learning,” in *2011 Ro-Man*, pp. 1–6, IEEE, 2011.
- [33] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.
- [34] R. Dearden, N. Friedman, and S. Russell, “Bayesian q-learning,” in *Aaai/iaai*, pp. 761–768, 1998.
- [35] C. Wirth, R. Akrou, G. Neumann, and J. Fürnkranz, “A survey of preference-based reinforcement learning methods,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 4945–4990, 2017.
- [36] C. Yu, T. Yang, W. Zhu, G. Li, *et al.*, “Learning shaping strategies in human-in-the-loop interactive reinforcement learning,” *arXiv preprint arXiv:1811.04272*, 2018.
- [37] Y. Wang, M. Huber, V. N. Papudesi, and D. J. Cook, “User-guided reinforcement learning of robot assistive tasks for an intelligent environment,” in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 1, pp. 424–429, IEEE, 2003.
- [38] K. Subramanian, C. L. Isbell Jr, and A. L. Thomaz, “Exploration from demonstration for interactive reinforcement learning,” in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 447–456, International Foundation for Autonomous Agents and Multiagent Systems, 2016.

-
- [39] E. Senft, P. Baxter, J. Kennedy, S. Lemaignan, and T. Belpaeme, "Supervised autonomy for online learning in human-robot interaction," *Pattern Recognition Letters*, vol. 99, pp. 77–86, 2017.
- [40] D. L. Moreno, C. V. Regueiro, R. Iglesias, and S. Barro, "Using prior knowledge to improve reinforcement learning in mobile robotics," 2004.
- [41] T. Mandel, Y.-E. Liu, E. Brunskill, and Z. Popović, "Where to add actions in human-in-the-loop reinforcement learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [42] A. Rosenfeld, M. Cohen, M. E. Taylor, and S. Kraus, "Leveraging human knowledge in tabular reinforcement learning: a study of human subjects," *The Knowledge Engineering Review*, vol. 33, 2018.
- [43] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in neural information processing systems*, pp. 1038–1044, 1996.
- [44] R. S. Sutton, A. G. Barto, et al., *Introduction to reinforcement learning*, vol. 135. MIT press Cambridge, 1998.
- [45] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.
- [46] W. B. Knox and P. Stone, "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*, pp. 5–12, International Foundation for Autonomous Agents and Multiagent Systems, 2010.
- [47] W. B. Knox and P. Stone, "Reinforcement learning from simultaneous human and mdp reward," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 475–482, International Foundation for Autonomous Agents and Multiagent Systems, 2012.
- [48] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [49] K. Asadi and M. L. Littman, "An alternative softmax operator for reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 243–252, JMLR. org, 2017.

A Transform State Features with Actions to State Action Features

Assume there are feature vectors f_s , that encode state s and have size N . f_s^n denotes the entry at position n . A linear model shall be learned for each action a in \mathcal{A} . Then those models can be represented by one matrix θ of size $|\mathcal{A}| \times N$, that is multiplied to the feature. The same can be expressed, when the features are constructed as

$$f_{s,a} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ f_s^0 \\ \vdots \\ f_s^N \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \left. \vphantom{\begin{pmatrix} 0 \\ \vdots \\ 0 \\ f_s^0 \\ \vdots \\ f_s^N \\ 0 \\ \vdots \\ 0 \end{pmatrix}} \right\} 0 \dots aN - 1 \\ \left. \vphantom{\begin{pmatrix} 0 \\ \vdots \\ 0 \\ f_s^0 \\ \vdots \\ f_s^N \\ 0 \\ \vdots \\ 0 \end{pmatrix}} \right\} aN \dots (a+1)N - 1 \\ \left. \vphantom{\begin{pmatrix} 0 \\ \vdots \\ 0 \\ f_s^0 \\ \vdots \\ f_s^N \\ 0 \\ \vdots \\ 0 \end{pmatrix}} \right\} (a+1)N \dots |\mathcal{A}|N - 1 \end{matrix}$$

and θ is just flattened to a row vector.

B User Study Forms

In the following the forms that were used for the user studies can be seen. Because the participants of the user studies spoke German, the used forms were written in German, too. Nevertheless, here are also the English translations given in Section B.1. The original versions can be found in Section B.2.

The title page and the general questions are the same for all user studies. Besides the final questions are only changed in the last user study with the taxi like gridworld. Due to that, only the relevant parts that changed compared to the forms before are shown here.

B.1 English User Study Forms

All used user study forms in English translation are added here.

Interactive Reinforcement Learning User Study



TECHNISCHE
UNIVERSITÄT
DARMSTADT

TU Darmstadt - Fachbereich Informatik - Hochschulstraße 10 - 64289 Darmstadt
Maximilian Kircher

Thank you very much for participating in this user study.

Please first read this document completely and fill out the questions on the next page. Please only start the experiment on the computer, when you were told to do so by the supervisor of the experiment. The acquired data is used for scientific purpose only. The data is confidential and it is used only in an anonymous way.

To ensure anonymity please insert the code you can see at the screen below in the field ID Code

ID Code:

Please fill out the following form:

Gender:

- male female diverse

Age:

- ≤17 18-20 21-29 30-39
 40-49 50-59 ≥60

What is your highest completed level of education?

- No Degree
 Lower Secondary School Leaving Certificate
 Intermediate Secondary School Leaving Certificate
 University Entrance Qualification
 Completed Vocational Training
 Bachelor
 Master
 PhD

Other Highest Completed Level of Education:

Do you know the term Reinforcement Learning?

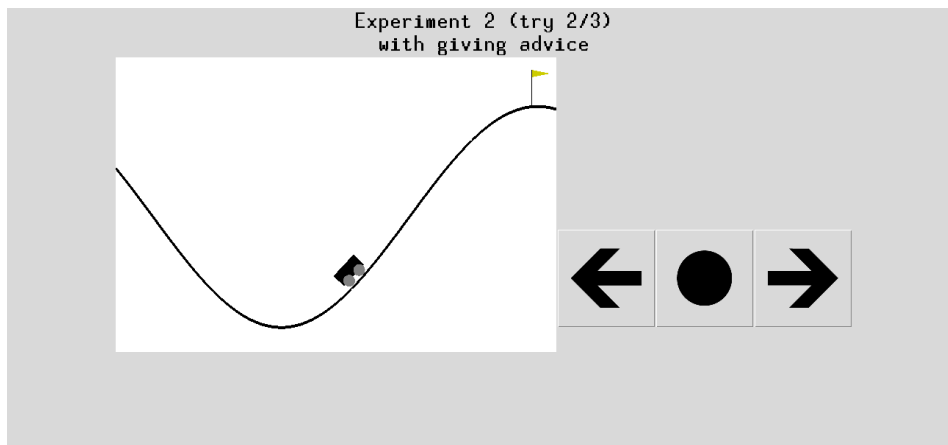
- No, never heard of it
 Heard of it, but I don't know what it is
 I know what Reinforcement Learning is

Please read the following instructions completely and attentively. After that please notify the supervisor of the experiment.

Giving Advice

In the experiment you will see a car in a valley between two hills (see below). The goal is that the car drives as fast as possible to the flag on the right hill on its own. To achieve this, the car can either accelerate to the left or to the right or do nothing. There is only one intensity of acceleration that can not be changed.

During the experiment you will see something similar to the following figure:



With the buttons on the right side you can show the car what you think would be good actions in the current situation. The arrows denote an acceleration in the corresponding direction. The dot stands for not accelerating. The car itself has always the full control what it is doing. So you can not directly control the car, but only show to it what you think it should do.

You can first try this out until the car reaches the flag or you quit the trying around, when you understood everything.

Please notify now the supervisor of the experiment.

Now the actual experiment starts. Here the car has only a limited amount of time to reach the flag. After that time it is reset to the middle of the valley. When the car reaches the flag, it is reset, too. Resetting the car does only change its position and velocity. The car does not forget what it has learned so far. After resetting the car 10 times one training run is finished. When a whole training run is finished, the car forgets everything it has learned.

Please notify again the supervisor of the experiment.

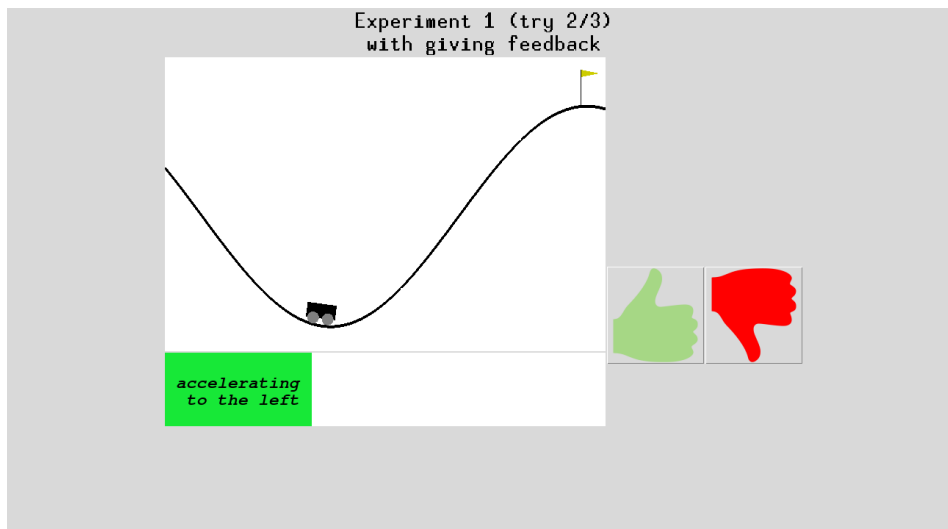
What was your impression at this experiment:

Please read the following instructions completely and attentively. After that please notify the supervisor of the experiment.

Giving Feedback

In the experiment you will see a car in a valley between two hills (see below). The goal is that the car drives as fast as possible to the flag on the right hill on its own. To achieve this, the car can either accelerate to the left or to the right or do nothing. There is only one intensity of acceleration that can not be changed.

During the experiment you will see something similar to the following figure:



With the buttons on the right side you can show the car, whether you think it is performing well in the current situation (thumb up) or not (thumb down). You can see the direction, that the car is currently accelerating to at the box under the car. This box visualizes the three possible actions of the car the following way:

- Box is green on the left side: Car is accelerating to the left
- Box is green in the middle: Car is not accelerating
- Box is green on the right side: Car is accelerating to the right

You can first try this out until the car reaches the flag or you quit the trying around, when you understood everything.

Please notify now the supervisor of the experiment.

Now the actual experiment starts. Here the car has only a limited amount of time to reach the flag. After that time it is reset to the middle of the valley. When the car reaches the flag, it is reset, too. Resetting the car does only change its position and velocity. The car does not forget what it has learned so far. After resetting the car 10 times one training run is finished. When a whole training run is finished, the car forgets everything it has learned.

Please notify again the supervisor of the experiment.

What was your impression at this experiment:

Final Questions

What kind of information was easier for you to provide, independent from the behavior of the car:

- It was easier to give feedback
- It was easier to advise actions
- Both were equally easy (difficult)

Which method led to the better results, according to your impression:

- By giving feedback
- By advising actions
- Both led to equally good (bad) results

Other Remarks:

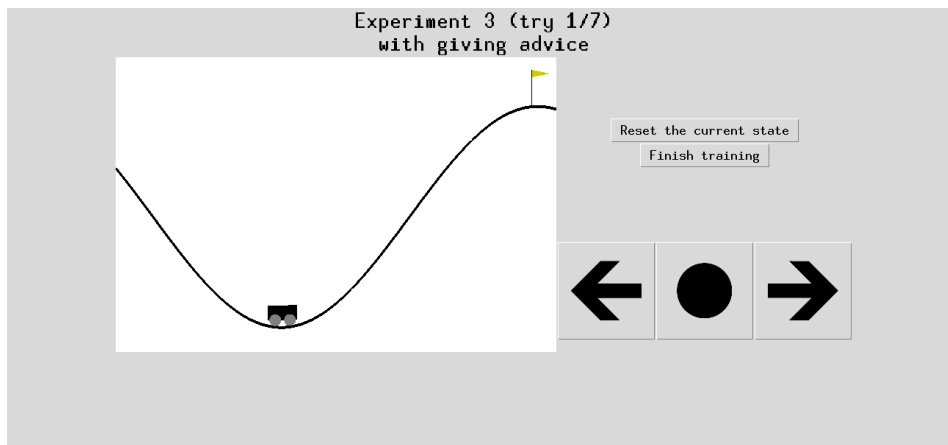
B.1.2 Mountain Car (2)

Please read the following instructions completely and attentively. After that please notify the supervisor of the experiment.

Giving Advice

In the experiment you will see a car in a valley between two hills (see below). The goal is that the car drives as fast as possible to the flag on the right hill on its own. To achieve this, the car can either accelerate to the left or to the right or do nothing. There is only one intensity of acceleration that can not be changed.

During the experiment you will see something similar to the following figure:



With the buttons on the right side you can show the car what you think would be good actions in the current situation. The arrows denote an acceleration in the corresponding direction. The dot stands for not accelerating. The car itself has always the full control what it is doing. So you can not directly control the car, but only show to it what you think it should do.

You can first try this out until the car reaches the flag or you quit the trying around, when you understood everything.

Please notify now the supervisor of the experiment.

Now the actual experiment starts. Here the car has only a limited amount of time to reach the flag. After that time it is reset to the middle of the valley. When the car reaches the flag, it is reset, too. Resetting the car does only change its position and velocity. The car does not forget what it has learned so far. You can also reset the car yourself with the button „Reset the current state“.

After resetting the car 7 times one training run is finished. When a whole training run is finished, the car forgets everything it has learned. You can also finish a training run yourself, if you think that the car has learned the desired behavior. To do that, press the button „Finish training“.

Please notify again the supervisor of the experiment.

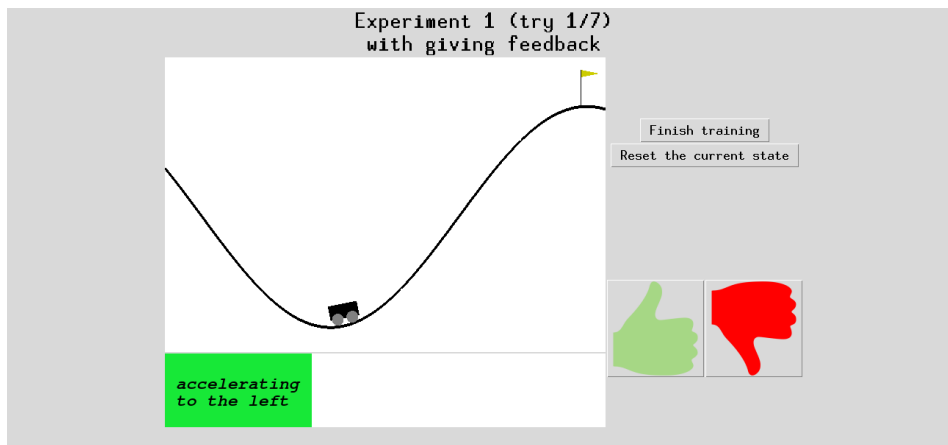
What was your impression at this experiment:

Please read the following instructions completely and attentively. After that please notify the supervisor of the experiment.

Giving Feedback

In the experiment you will see a car in a valley between two hills (see below). The goal is that the car drives as fast as possible to the flag on the right hill on its own. To achieve this, the car can either accelerate to the left or to the right or do nothing. There is only one intensity of acceleration that can not be changed.

During the experiment you will see something similar to the following figure:



With the buttons on the right side you can show the car, whether you think it is performing well in the current situation (thumb up) or not (thumb down). You can see the direction, that the car is currently accelerating to at the box under the car. This box visualizes the three possible actions of the car the following way:

- Box is green on the left side: Car is accelerating to the left
- Box is green in the middle: Car is not accelerating
- Box is green on the right side: Car is accelerating to the right

You can first try this out until the car reaches the flag or you quit the trying around, when you understood everything.

Please notify now the supervisor of the experiment.

Now the actual experiment starts. Here the car has only a limited amount of time to reach the flag. After that time it is reset to the middle of the valley. When the car reaches the flag, it is reset, too. Resetting the car does only change its position and velocity. The car does not forget what it has learned so far. You can also reset the car yourself with the button „Reset the current state“.

After resetting the car 7 times one training run is finished. When a whole training run is finished, the car forgets everything it has learned. You can also finish a training run yourself, if you think that the car has learned the desired behavior. To do that, press the button „Finish training“.

Please notify again the supervisor of the experiment.

What was your impression at this experiment:

B.1.3 Lunar Lander

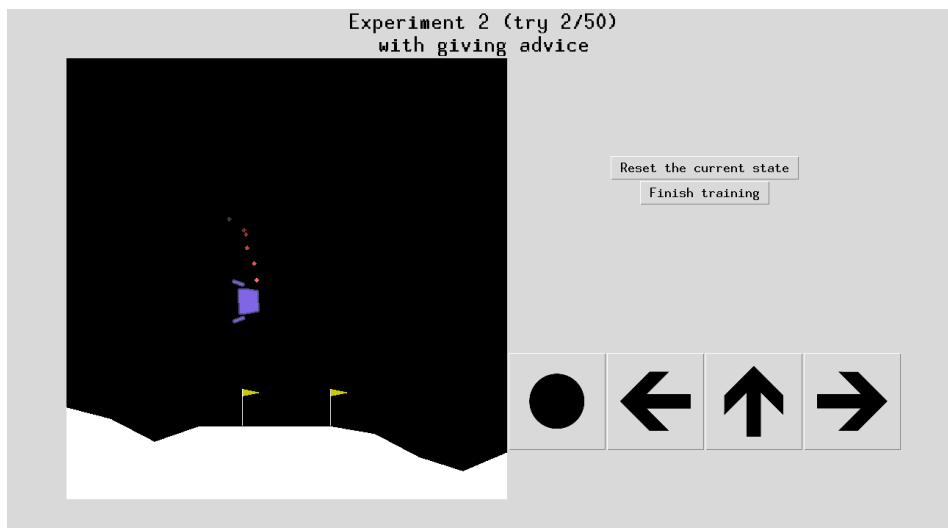
Please read the following instructions completely and attentively. After that please notify the supervisor of the experiment.

Giving Advice

In this experiment, you will see a spaceship that flies above the surface of the moon (see below). The goal is that the spaceship lands as safely as possible autonomously on the moon. It is especially good, if it lands between the two flags in the middle of the screen.

The spaceship has three engines to achieve this. When one of them is active, this can be seen at the orange particles that are ejected then. The left engine slightly moves the spaceship to the right and rotates it clockwise. The right engine has the exact opposite effect. The central engine accelerates the spaceship to the top. All those accelerations are relative to the orientation of the spaceship. That is, for example, if the spaceship is upside down, the central engine accelerates it towards the surface of the moon. Without any acceleration, it falls downward.

During the experiment you will see something similar to the following figure:



With the buttons on the right side, you can show the spaceship what you consider to be a good action in the current situation. The arrow pointing upwards stands for the central engine, the arrow to the left for the right one and, accordingly, the arrow to the right for the left engine. The dot stands for not using any engine. The spaceship always flies on its own. Thus, you cannot directly control it, but show to it what you think it should do. You can now try this out until the spaceship reaches the surface of the moon.

Please notify now the supervisor of the experiment.

Now the actual experiment begins. Here the spaceship has only a limited amount of time to land. After the expiration of this time it is reset again to a random position above the moon. When the spaceship landed already successfully or it crashed down or left the screen to the left or to the right, it is reset, too. Resetting the spaceship only changes its position and velocity. It does not forget what it has learned so far. You can also reset the spaceship yourself with the button „Reset the current state“.

Whether a trial was successful is shown to you afterwards („Success“/ „Failure“).

After the spaceship was reset 50 times, one training run is finished. When a whole training run is finished, the spaceship forgets everything it has learned. You can also finish a training run yourself, when you think that the spaceship has learned the desired behavior. To do that press the button „Finish training“.

Please notify again the supervisor of the experiment.

What was your impression at this experiment:

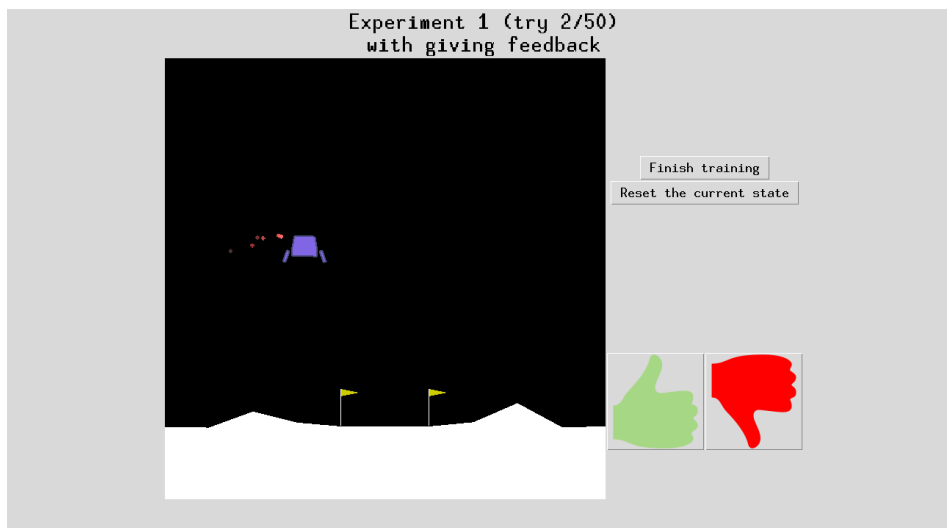
Please read the following instructions completely and attentively. After that please notify the supervisor of the experiment.

Giving Feedback

In this experiment, you will see a spaceship that flies above the surface of the moon (see below). The goal is that the spaceship lands as safely as possible autonomously on the moon. It is especially good, if it lands between the two flags in the middle of the screen.

The spaceship has three engines to achieve this. When one of them is active, this can be seen at the orange particles that are ejected then. The left engine slightly moves the spaceship to the right and rotates it clockwise. The right engine has the exact opposite effect. The central engine accelerates the spaceship to the top. All those accelerations are relative to the orientation of the spaceship. That is, for example, if the spaceship is upside down, the central engine accelerates it towards the surface of the moon. Without any acceleration, it falls downward.

During the experiment you will see something similar to the following figure:



With the buttons on the right side, you can show to the spaceship, whether it is behaving well in the current situation (thumb up) or not (thumb down).

You can now try this out until the spaceship reaches the surface of the moon.

Please notify now the supervisor of the experiment.

Now the actual experiment begins. Here the spaceship has only a limited amount of time to land. After the expiration of this time it is reset again to a random position above the moon. When the spaceship landed already successfully or it crashed down or left the screen to the left or to the right, it is reset, too. Resetting the spaceship only changes its position and velocity. It does not forget what it has learned so far. You can also reset the spaceship yourself with the button „Reset the current state“.

Whether a trial was successful is shown to you afterwards („Success“/ „Failure“).

After the spaceship was reset 50 times, one training run is finished. When a whole training run is finished, the spaceship forgets everything it has learned. You can also finish a training run yourself, when you think that the spaceship has learned the desired behavior. To do that press the button „Finish training“.

Please notify again the supervisor of the experiment.

What was your impression at this experiment:

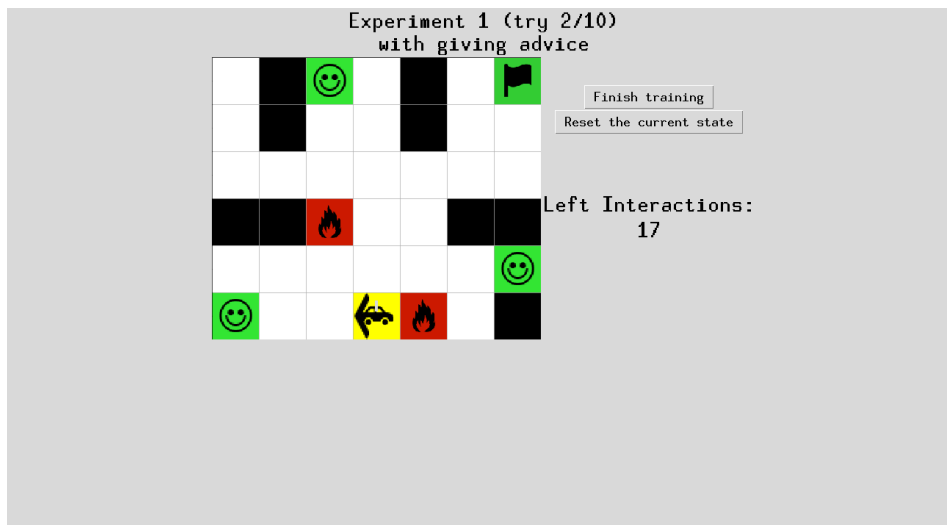
B.1.4 Taxi Like Gridworld

Please read the following instructions completely and attentively. After that please notify the supervisor of the experiment.

Giving Advice

The goal of this experiment is that a taxi (yellow car) first collects as fast as possible autonomously all passengers (green smileys) and drives them to the goal (green flag), afterwards. All white fields can be used to do this. The black fields are walls. It does not harm to drive against them, but the taxi can not get through them. Furthermore, there are two fires, who destroy the taxi, if it drives into them. This should be avoided.

During the experiment you will see something similar to the following figure:



The taxi can move at each time step exactly one field to the right, left, up, or down. Arrows around the taxi show, what action it plans to perform next. With the arrow keys of your keyboard you can show to the taxi what you consider to be a good action at the current field. When you proposed an action, it is performed in the next step. With the space key you can avoid that the currently predicted action is performed. You can only advice or prevent actions 20 times. On the right side you can see how many interactions you have left. You can now try this out until the taxi has reached the flag, drove into a fire or you, when you understood everything, abort the trying out.

Please notify now the supervisor of the experiment.

Now the actual experiment begins. Here the taxi has only a limited (but quite long) amount of time to bring the passengers to the flag. After that it is reset to a random position and loses all passengers that it may have collected already. When the taxi reaches the flag (also without passengers) or drives into a fire, it is reset, too. With each reset the left number of advices to the taxi are also reset to 20. Doing that, the taxi does not forget what it has learned. You can also reset the taxi yourself with the button „Reset the current state“.

After the taxi was reset 12 times, one training run is finished. When a whole training run is finished, the taxi forgets everything it has learned. You can also finish a training run yourself, when you think that the taxi has learned the desired behavior. Use the button „Finish training“for that.

Please notify again the supervisor of the experiment.

What was your impression with this experiment:

Now the same experiment is repeated with the only difference that the taxi drives a bit faster this time.

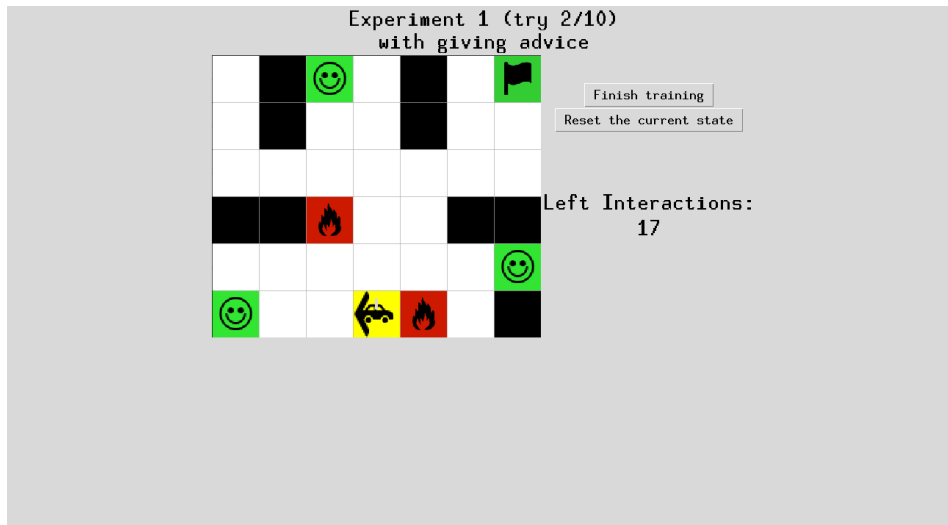
What was your impression with this experiment:

Please read the following instructions completely and attentively. After that please notify the supervisor of the experiment.

Giving Feedback

The goal of this experiment is that a taxi (yellow car) first collects as fast as possible autonomously all passengers (green smileys) and drives them to the goal (green flag), afterwards. All white fields can be used to do this. The black fields are walls. It does not harm to drive against them, but the taxi can not get through them. Furthermore, there are two fires, who destroy the taxi, if it drives into them. This should be avoided.

During the experiment you will see something similar to the following figure:



The taxi can move at each time step exactly one field to the right, left, up, or down. Arrows around the taxi show, what action it plans to perform next. You can judge the currently displayed direction with your keyboard. To do this, the arrow key that points to the top and the plus key stand for a positive feedback, while the arrow key that points downwards and the minus key stand for negative feedback.

You can only judge the taxi 20 times. On the right side you can see how many interactions you have left.

You can now try this out until the taxi has reached the flag, drove into a fire or you, when you understood everything, abort the trying out.

Please notify now the supervisor of the experiment.

Now the actual experiment begins. Here the taxi has only a limited (but quite long) amount of time to bring the passengers to the flag. After that it is reset to a random position and loses all passengers that it may have collected already. When the taxi reaches the flag (also without passengers) or drives into a fire, it is reset, too. With each reset the left number of advices to the taxi are also reset to 20. Doing that, the taxi does not forget what it has learned. You can also reset the taxi yourself with the button „Reset the current state“.

After the taxi was reset 12 times, one training run is finished. When a whole training run is finished, the taxi forgets everything it has learned. You can also finish a training run yourself, when you think that the taxi has learned the desired behavior. Use the button „Finish training“for that.

Please notify again the supervisor of the experiment.

What was your impression with this experiment:

Now the same experiment is repeated with the only difference that the taxi drives a bit faster this time.

What was your impression with this experiment:

Abschließende Fragen

What kind of information was easier for you to provide, independent from the behavior of the taxi:

- It was easier to give feedback
- It was easier to advise actions
- Both were equally easy (difficult)

Which method led to the better results, according to your impression:

- By giving feedback
- By advising actions
- Both led to equally good (bad) results

What speed of the taxi was more pleasant for you to give feedback:

- The fast taxi
- The slow taxi
- Both were similarly pleasant

What speed of the taxi was more pleasant for you to give advice:

- The fast taxi
- The slow taxi
- Both were similarly pleasant

Other Remarks:

B.2 German User Study Forms

Here all used user study forms in their original German version can be found.

Interactive Reinforcement Learning User Study



TECHNISCHE
UNIVERSITÄT
DARMSTADT

TU Darmstadt - Fachbereich Informatik - Hochschulstraße 10 - 64289 Darmstadt
Maximilian Kircher

Vielen Dank für Ihre Bereitschaft an dieser Studie teilzunehmen.

Bitte lesen Sie sich zunächst dieses Dokument vollständig durch und beantworten Sie die Fragen auf der nächsten Seite. Bitte starten Sie das Experiment am Computer erst, wenn Sie vom Betreuer des Experiments dazu aufgefordert wurden.

Die gewonnenen Daten dienen ausschließlich wissenschaftlichen Zwecken. Ihre Daten sind vertraulich und werden nur in anonymisierter Form genutzt.

Um die Anonymität ihrer Daten sicherzustellen bitten wir Sie den Code, den Sie auf dem Bildschirm sehen unten im Feld ID Code einzutragen.

ID Code:

Bitte füllen Sie folgende Angaben aus:

Geschlecht:

- männlich weiblich divers

Alter:

- ≤17 18-20 21-29 30-39
 40-49 50-59 ≥60

Was ist Ihr höchster Bildungsabschluss?

- Kein Schulabschluss
 Grund-/Hauptschulabschluss
 Realschulabschluss
 Abitur
 Abgeschlossene Ausbildung
 Bachelor
 Master oder Diplom
 Doktor

Anderer Höchster Bildungsabschluss:

Kennen Sie den Begriff Reinforcement Learning?

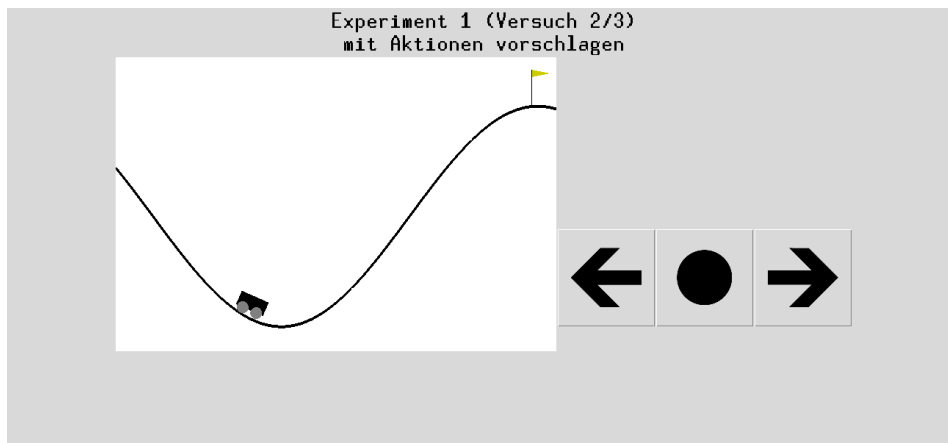
- Nein, noch nie gehört
 Schonmal gehört, aber ich weiß nicht, was es ist
 Ich weiß was Reinforcement Learning ist

Bitte lesen Sie sich die folgenden Instruktionen aufmerksam und vollständig durch, und melden Sie sich dann bei dem Betreuer des Experiments.

Aktionen vorschlagen

In dem Experiment, werden Sie ein Auto in einem Tal zwischen zwei Hügeln sehen (siehe unten). Das Ziel ist es, dass das Auto möglichst schnell selbstständig zu der Fahne auf dem rechten Hügel fährt. Um das zu erreichen, kann das Auto entweder nach links oder rechts beschleunigen, oder nichts tun. Es gibt dabei nur eine Intensität der Beschleunigung, die nicht geändert werden kann.

Sie werden während des Experiments eine Darstellung, ähnlich der folgenden Abbildung, sehen:



Mit den Knöpfen auf der rechten Seite können Sie dem Auto anzeigen, was Sie in der aktuellen Situation für eine gute Aktion halten. Die Pfeile stehen dabei für eine Beschleunigung in die entsprechende Richtung, der Punkt steht dafür nicht zu beschleunigen. Das Auto fährt immer selbstständig. Sie können es also nicht direkt steuern, sondern ihm nur zeigen was es Ihrer Meinung nach tun sollte.

Sie können dies zunächst ausprobieren bis das Auto die Fahne erreicht hat, oder Sie, wenn Sie alles verstanden haben, das Ausprobieren abbrechen.

Bitte melden Sie sich jetzt bei dem Betreuer des Experiments.

Nun beginnt das eigentliche Experiment. Hier hat das Auto nur eine begrenzte Zeit um die Fahne zu erreichen. Danach wird es wieder in das Tal zurückgesetzt. Wenn das Auto die Fahne erreicht hat, wird es ebenfalls ins Tal zurückgesetzt. Beim Zurücksetzen werden allerdings nur Position und Geschwindigkeit geändert. Das Auto vergisst dabei nicht, was es bisher gelernt hat.

Nachdem das Auto 10 mal zurückgesetzt wurde ist ein Trainingsdurchlauf beendet. Wenn ein ganzer Trainingsdurchlauf beendet ist, vergisst das Auto alles Erlernete.

Bitte melden Sie sich jetzt erneut bei dem Betreuer des Experiments.

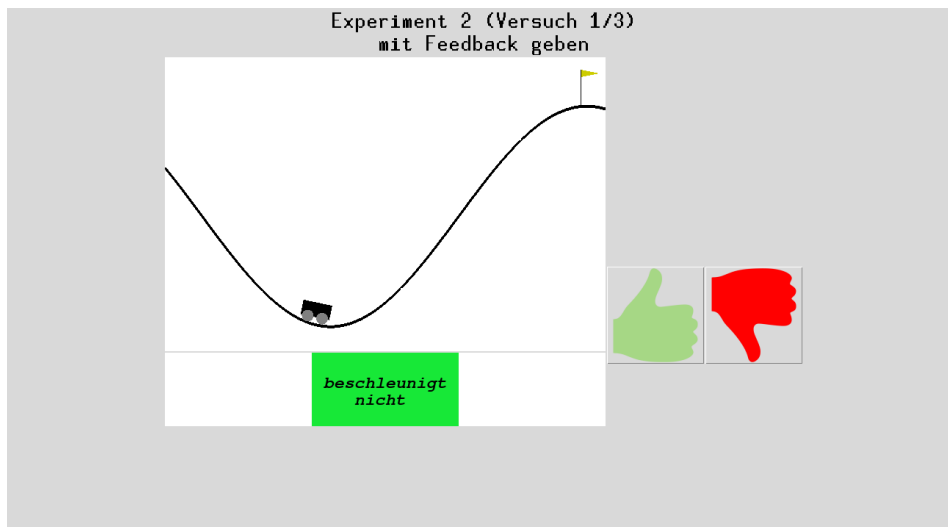
Was war Ihr Eindruck bei diesem Experiment:

Bitte lesen Sie sich die folgenden Instruktionen aufmerksam und vollständig durch, und melden Sie sich dann bei dem Betreuer des Experiments.

Aktionen bewerten

In dem Experiment, werden Sie ein Auto in einem Tal zwischen zwei Hügeln sehen (siehe unten). Das Ziel ist es, dass das Auto möglichst schnell selbstständig zu der Fahne auf dem rechten Hügel fährt. Um das zu erreichen kann das Auto entweder nach links oder rechts beschleunigen, oder nichts tun. Es gibt dabei nur eine Intensität der Beschleunigung, die nicht geändert werden kann.

Sie werden während des Experiments eine Darstellung, ähnlich der folgenden Abbildung, sehen:



Mit den Knöpfen auf der rechten Seite können Sie dem Auto anzeigen, ob es sich in der aktuellen Situation gut verhält (Daumen hoch) oder nicht (Daumen runter). In welche Richtung das Auto gerade beschleunigt können Sie an dem Kasten unter dem Auto erkennen. Dieser visualisiert die drei möglichen Aktionen folgendermaßen:

- Kasten ist links grün: Auto beschleunigt nach links
- Kasten ist mittig grün: Auto beschleunigt nicht
- Kasten ist rechts grün: Auto beschleunigt nach rechts

Sie können dies zunächst ausprobieren bis das Auto die Fahne erreicht hat, oder Sie, wenn Sie alles verstanden haben, das Ausprobieren abbrechen.

Bitte melden Sie sich jetzt bei dem Betreuer des Experiments.

Nun beginnt das eigentliche Experiment. Hier hat das Auto nur eine begrenzte Zeit um die Fahne zu erreichen. Danach wird es wieder in das Tal zurückgesetzt. Wenn das Auto die Fahne erreicht hat, wird es ebenfalls ins Tal zurückgesetzt. Beim Zurücksetzen werden allerdings nur Position und Geschwindigkeit geändert. Das Auto vergisst dabei nicht, was es bisher gelernt hat.

Nachdem das Auto 10 mal zurückgesetzt wurde ist ein Trainingsdurchlauf beendet. Wenn ein ganzer Trainingsdurchlauf beendet ist, vergisst das Auto alles Erlernete.

Bitte melden Sie sich jetzt erneut bei dem Betreuer des Experiments.

Was war Ihr Eindruck bei diesem Experiment:

Abschließende Fragen

Welche Art der Information war für Sie einfacher zu geben, unabhängig vom Verhalten des Autos:

- Es war einfacher Feedback zu geben
- Es war einfacher Aktionen vorzuschlagen
- Beides war gleich einfach (schwer)

Welche Methode hat nach ihrem Eindruck zu den besseren Ergebnissen geführt:

- Durch Feedback
- Durch vorzuschlagen von Aktionen
- Beides führte zu gleich guten (schlechten) Ergebnissen

Sonstige Anmerkungen:

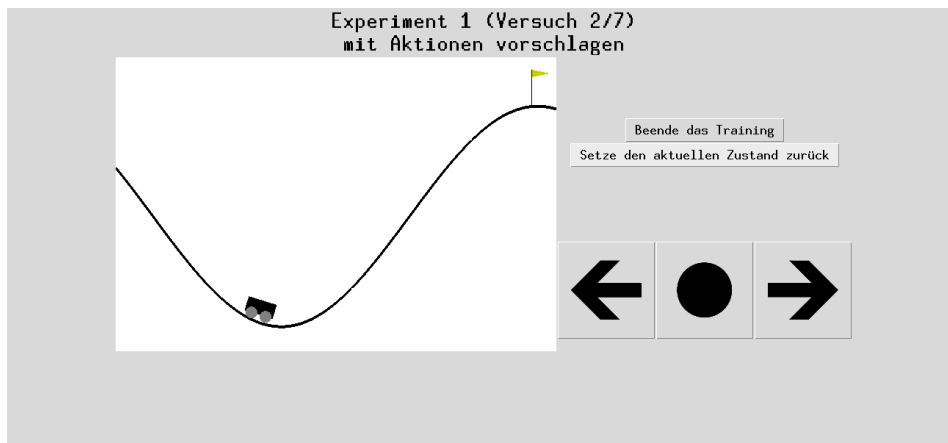
B.2.2 Mountain Car (2)

Bitte lesen Sie sich die folgenden Instruktionen aufmerksam und vollständig durch, und melden Sie sich dann bei dem Betreuer des Experiments.

Aktionen vorschlagen

In dem Experiment, werden Sie ein Auto in einem Tal zwischen zwei Hügeln sehen (siehe unten). Das Ziel ist es, dass das Auto möglichst schnell selbstständig zu der Fahne auf dem rechten Hügel fährt. Um das zu erreichen, kann das Auto entweder nach links oder rechts beschleunigen, oder nichts tun. Es gibt dabei nur eine Intensität der Beschleunigung, die nicht geändert werden kann.

Sie werden während des Experiments eine Darstellung, ähnlich der folgenden Abbildung, sehen:



Mit den Knöpfen auf der rechten Seite können Sie dem Auto anzeigen, was Sie in der aktuellen Situation für eine gute Aktion halten. Die Pfeile stehen dabei für eine Beschleunigung in die entsprechende Richtung, der Punkt steht dafür nicht zu beschleunigen. Das Auto fährt immer selbstständig. Sie können es also nicht direkt steuern, sondern ihm nur zeigen was es Ihrer Meinung nach tun sollte.

Sie können dies zunächst ausprobieren bis das Auto die Fahne erreicht hat, oder Sie, wenn Sie alles verstanden haben, das Ausprobieren abbrechen.

Bitte melden Sie sich jetzt bei dem Betreuer des Experiments.

Nun beginnt das eigentliche Experiment. Hier hat das Auto nur eine begrenzte Zeit um die Fahne zu erreichen. Danach wird es wieder in das Tal zurückgesetzt. Wenn das Auto die Fahne erreicht hat, wird es ebenfalls ins Tal zurückgesetzt. Beim Zurücksetzen werden allerdings nur Position und Geschwindigkeit geändert. Das Auto vergisst dabei nicht, was es bisher gelernt hat. Sie können das Auto auch selbst mit dem Knopf „Setze den aktuellen Zustand zurück“ zurücksetzen.

Nachdem das Auto 7 mal zurückgesetzt wurde ist ein Trainingsdurchlauf beendet. Wenn ein ganzer Trainingsdurchlauf beendet ist, vergisst das Auto alles Erlernte. Sie können einen Trainingsdurchlauf auch selbst beenden, wenn Sie der Meinung sind, dass das Auto das gewünschte Verhalten erlernt hat. Hierzu drücken Sie den Knopf „Beende das Training“.

Bitte melden Sie sich jetzt erneut bei dem Betreuer des Experiments.

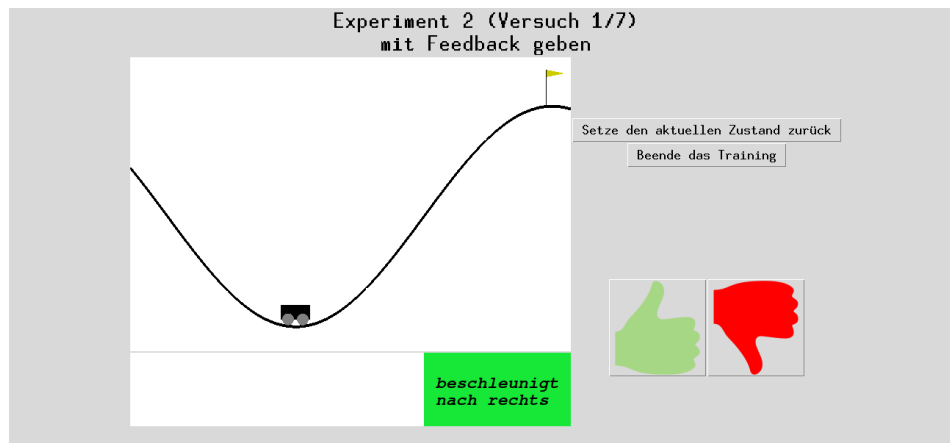
Was war Ihr Eindruck bei diesem Experiment:

Bitte lesen Sie sich die folgenden Instruktionen aufmerksam und vollständig durch, und melden Sie sich dann bei dem Betreuer des Experiments.

Aktionen bewerten

In dem Experiment, werden Sie ein Auto in einem Tal zwischen zwei Hügeln sehen (siehe unten). Das Ziel ist es, dass das Auto möglichst schnell selbstständig zu der Fahne auf dem rechten Hügel fährt. Um das zu erreichen kann das Auto entweder nach links oder rechts beschleunigen, oder nichts tun. Es gibt dabei nur eine Intensität der Beschleunigung, die nicht geändert werden kann.

Sie werden während des Experiments eine Darstellung, ähnlich der folgenden Abbildung, sehen:



Mit den Knöpfen auf der rechten Seite können Sie dem Auto anzeigen, ob es sich in der aktuellen Situation gut verhält (Daumen hoch) oder nicht (Daumen runter). In welche Richtung das Auto gerade beschleunigt können Sie an dem Kasten unter dem Auto erkennen. Dieser visualisiert die drei möglichen Aktionen folgendermaßen:

- Kasten ist links grün: Auto beschleunigt nach links
- Kasten ist mittig grün: Auto beschleunigt nicht
- Kasten ist rechts grün: Auto beschleunigt nach rechts

Sie können dies zunächst ausprobieren bis das Auto die Fahne erreicht hat, oder Sie, wenn Sie alles verstanden haben, das Ausprobieren abbrechen.

Bitte melden Sie sich jetzt bei dem Betreuer des Experiments.

Nun beginnt das eigentliche Experiment. Hier hat das Auto nur eine begrenzte Zeit um die Fahne zu erreichen. Danach wird es wieder in das Tal zurückgesetzt. Wenn das Auto die Fahne erreicht hat, wird es ebenfalls ins Tal zurückgesetzt. Beim Zurücksetzen werden allerdings nur Position und Geschwindigkeit geändert. Das Auto vergisst dabei nicht, was es bisher gelernt hat. Sie können das Auto auch selbst mit dem Knopf „Setze den aktuellen Zustand zurück“ zurücksetzen.

Nachdem das Auto 7 mal zurückgesetzt wurde ist ein Trainingsdurchlauf beendet. Wenn ein ganzer Trainingsdurchlauf beendet ist, vergisst das Auto alles Erlernte. Sie können einen Trainingsdurchlauf auch selbst beenden, wenn Sie der Meinung sind, dass das Auto das gewünschte Verhalten erlernt hat. Hierzu drücken Sie den Knopf „Beende das Training“.

Bitte melden Sie sich jetzt erneut bei dem Betreuer des Experiments.

Was war Ihr Eindruck bei diesem Experiment:

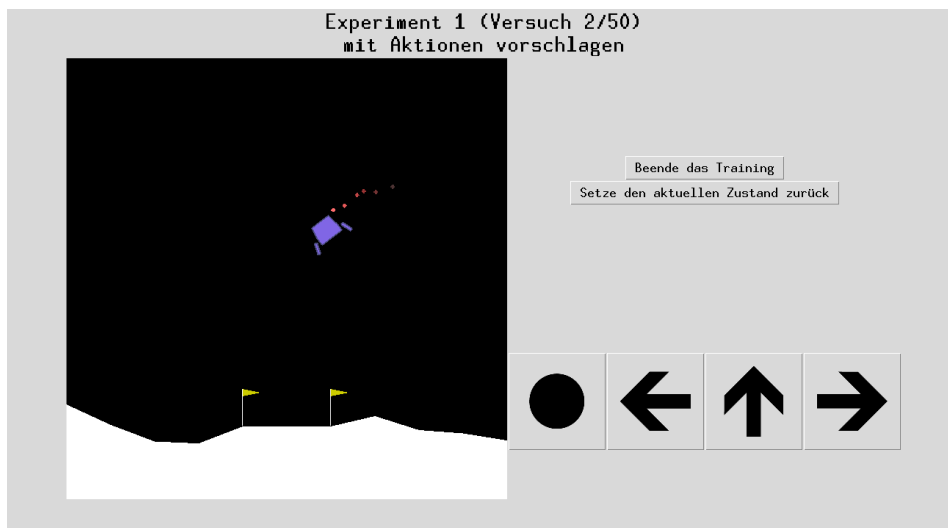
Bitte lesen Sie sich die folgenden Instruktionen aufmerksam und vollständig durch, und melden Sie sich dann bei dem Betreuer des Experiments.

Aktionen vorschlagen

In dem Experiment werden Sie ein Raumschiff, das über der Mondoberfläche fliegt, sehen (siehe unten). Das Ziel ist es, dass das Raumschiff möglichst sicher selbstständig auf dem Mond landet. Am besten ist es, wenn es zwischen den beiden Fahnen in der Mitte des Bildschirms landet.

Um das zu erreichen, stehen dem Raumschiff drei Antriebe zur Verfügung. Wenn einer dieser Antriebe aktiv ist, ist dies daran erkennbar, dass orangene Partikel vom Raumschiff abgestoßen werden. Der linke Antrieb bewegt das Raumschiff leicht nach rechts und dreht es im Uhrzeigersinn. Der rechte Antrieb hat den genau gegenteiligen Effekt. Der zentrale Antrieb beschleunigt das Raumschiff nach oben. Alle diese Beschleunigungen wirken relativ zur Ausrichtung des Raumschiffs. Das heißt beispielsweise, wenn das Raumschiff auf dem Kopf steht (die Füße zeigen nach oben) bewirkt der zentrale Antrieb eine Beschleunigung in Richtung der Mondoberfläche. Ohne Beschleunigung fällt das Raumschiff herunter.

Sie werden während des Experiments eine Darstellung, ähnlich der folgenden Abbildung, sehen:



Mit den Knöpfen auf der rechten Seite können Sie dem Raumschiff anzeigen, was Sie in der aktuellen Situation für eine gute Aktion halten. Der Pfeil nach oben steht für den zentralen, der Pfeil nach links für den rechten und der Pfeil nach rechts dementsprechend für den linken Antrieb. Der Punkt steht dafür keinen Antrieb zu verwenden. Das Raumschiff fliegt immer selbstständig. Sie können es also nicht direkt steuern, sondern ihm nur zeigen was es Ihrer Meinung nach tun sollte.

Sie können dies zunächst ausprobieren bis das Raumschiff die Mondoberfläche erreicht hat.

Bitte melden Sie sich jetzt bei dem Betreuer des Experiments.

Nun beginnt das eigentliche Experiment. Hier hat das Raumschiff nur eine begrenzte Zeit um zu landen. Nach Ablauf dieser Zeit wird es wieder an eine zufällige Position über dem Mond zurückgesetzt. Wenn das Raumschiff zuvor erfolgreich gelandet ist oder es abgestürzt ist, beziehungsweise den Bildschirm nach links oder rechts verlassen hat, wird es ebenfalls zurückgesetzt. Beim Zurücksetzen werden allerdings nur Position und Geschwindigkeit geändert. Das Raumschiff vergisst dabei nicht, was es bisher gelernt hat. Sie können das Raumschiff auch selbst mit dem Knopf „Setze den aktuellen Zustand zurück“ zurücksetzen.

Ob ein Versuch erfolgreich war wird Ihnen nach dem Versuch angezeigt („Erfolg“/ „Gescheitert“).

Nachdem das Raumschiff 50 mal zurückgesetzt wurde ist ein Trainingsdurchlauf beendet. Wenn ein ganzer Trainingsdurchlauf beendet ist, vergisst das Raumschiff alles Erlernte. Sie können einen Trainingsdurchlauf auch selbst beenden, wenn Sie der Meinung sind, dass das Raumschiff das gewünschte Verhalten erlernt hat. Hierzu drücken Sie den Knopf „Beende das Training“.

Bitte melden Sie sich jetzt erneut bei dem Betreuer des Experiments.

Was war Ihr Eindruck bei diesem Experiment:

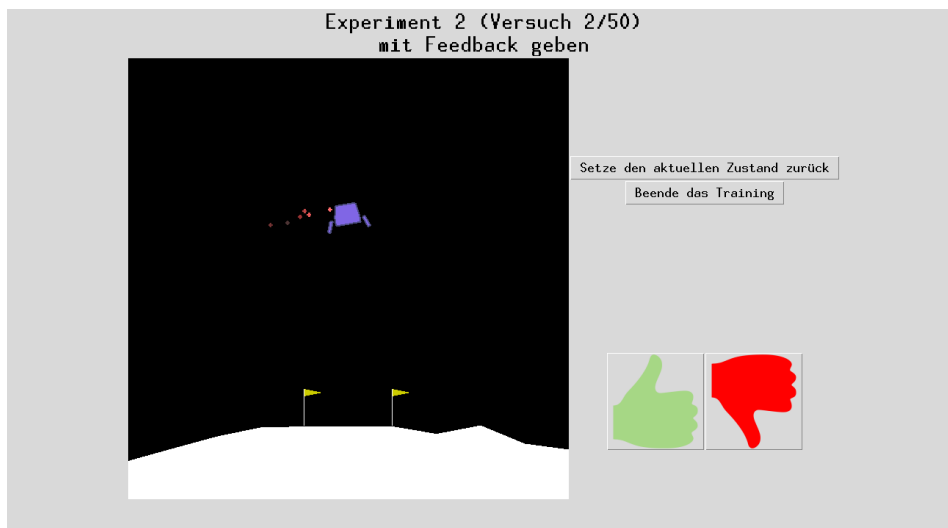
Bitte lesen Sie sich die folgenden Instruktionen aufmerksam und vollständig durch, und melden Sie sich dann bei dem Betreuer des Experiments.

Aktionen bewerten

In dem Experiment werden Sie ein Raumschiff, das über der Mondoberfläche fliegt, sehen (siehe unten). Das Ziel ist es, dass das Raumschiff möglichst sicher selbstständig auf dem Mond landet. Am besten ist es, wenn es zwischen den beiden Fahnen in der Mitte des Bildschirms landet.

Um das zu erreichen, stehen dem Raumschiff drei Antriebe zur Verfügung. Wenn einer dieser Antriebe aktiv ist, ist dies daran erkennbar, dass orangene Partikel vom Raumschiff abgestoßen werden. Der linke Antrieb bewegt das Raumschiff leicht nach rechts und dreht es im Uhrzeigersinn. Der rechte Antrieb hat den genau gegenteiligen Effekt. Der zentrale Antrieb beschleunigt das Raumschiff nach oben. Alle diese Beschleunigungen wirken relativ zur Ausrichtung des Raumschiffs. Das heißt beispielsweise, wenn das Raumschiff auf dem Kopf steht (die Füße zeigen nach oben) bewirkt der zentrale Antrieb eine Beschleunigung in Richtung der Mondoberfläche. Ohne Beschleunigung fällt das Raumschiff herunter.

Sie werden während des Experiments eine Darstellung, ähnlich der folgenden Abbildung, sehen:



Mit den Knöpfen auf der rechten Seite können Sie dem Raumschiff anzeigen, ob es sich in der aktuellen Situation gut verhält (Daumen hoch) oder nicht (Daumen runter).

Sie können dies zunächst ausprobieren bis das Raumschiff die Mondoberfläche erreicht hat.

Bitte melden Sie sich jetzt bei dem Betreuer des Experiments.

Nun beginnt das eigentliche Experiment. Hier hat das Raumschiff nur eine begrenzte Zeit um zu landen. Nach Ablauf dieser Zeit wird es wieder an eine zufällige Position über dem Mond zurückgesetzt. Wenn das Raumschiff zuvor erfolgreich gelandet ist oder es abgestürzt ist, beziehungsweise den Bildschirm nach links oder rechts verlassen hat, wird es ebenfalls zurückgesetzt. Beim Zurücksetzen werden allerdings nur Position und Geschwindigkeit geändert. Das Raumschiff vergisst dabei nicht, was es bisher gelernt hat. Sie können das Raumschiff auch selbst mit dem Knopf „Setze den aktuellen Zustand zurück“ zurücksetzen.

Ob ein Versuch erfolgreich war wird Ihnen nach dem Versuch angezeigt („Erfolg“/ „Gescheitert“).

Nachdem das Raumschiff 50 mal zurückgesetzt wurde ist ein Trainingsdurchlauf beendet. Wenn ein ganzer Trainingsdurchlauf beendet ist, vergisst das raumschiff alles Erlernte. Sie können einen Trainingsdurchlauf auch selbst beenden, wenn Sie der Meinung sind, dass das Raumschiff das gewünschte Verhalten erlernt hat. Hierzu drücken Sie den Knopf „Beende das Training“.

Bitte melden Sie sich jetzt erneut bei dem Betreuer des Experiments.

Was war Ihr Eindruck bei diesem Experiment:

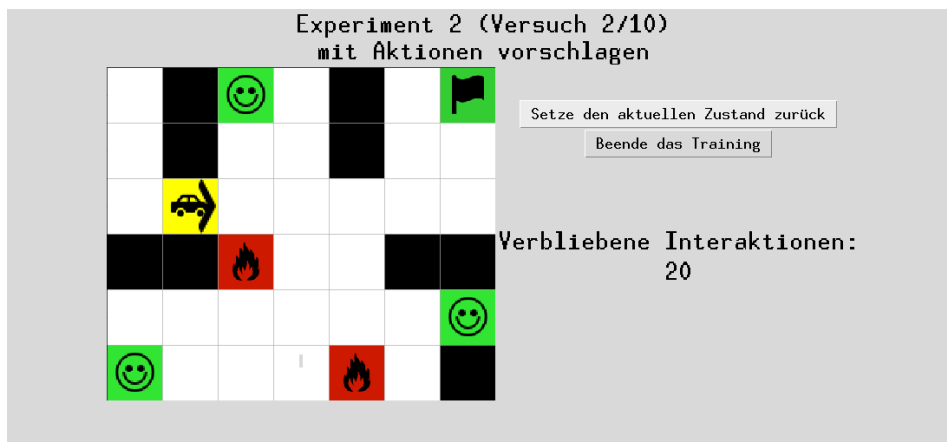
B.2.4 Taxi Like Gridworld

Bitte lesen Sie sich die folgenden Instruktionen aufmerksam und vollständig durch, und melden Sie sich dann bei dem Betreuer des Experiments.

Aktionen vorschlagen

Das Ziel dieses Experiments ist, dass ein Taxi (gelbes Auto) möglichst schnell selbstständig zunächst alle Passagiere (grüne Smileys) einsammelt und diese dann zum Ziel (grüne Fahne) fährt. Dabei können alle weißen Felder verwendet werden. Die schwarzen Felder sind Mauern. Es schadet nicht gegen diese zu fahren, aber das Taxi kann sie nicht durchdringen. Darüber hinaus gibt es zwei Feuer, die das Taxi, wenn es in eines von diesen fährt, zerstören. Dies sollte vermieden werden.

Sie werden während des Experiments eine Darstellung, ähnlich der folgenden Abbildung, sehen:



Das Taxi kann sich zu jedem Zeitschritt um genau ein Feld nach rechts, links, oben oder unten bewegen. Mit Pfeilen um das Taxi wird angezeigt, welche Aktion es plant als nächstes auszuführen. Mit den Pfeiltasten der Tastatur können Sie dem Taxi anzeigen, was Sie in der aktuellen Situation für eine gute Aktion halten. Wenn Sie eine Aktion vorgeschlagen haben, wird diese im nächsten Schritt ausgeführt. Mit der Leertaste können Sie verhindern, dass der aktuell vorhergesagte Schritt ausgeführt wird. Sie können dem Taxi nur 20 mal Aktionen vorschlagen oder Aktionen verhindern. Auf der rechten Seite wird Ihnen angezeigt, wie viele Male Ihnen noch verbleiben.

Sie können dies zunächst ausprobieren bis das Taxi die Fahne erreicht hat, in ein Feuer gefahren ist, oder Sie, wenn Sie alles verstanden haben, das Ausprobieren abbrechen.

Bitte melden Sie sich jetzt bei dem Betreuer des Experiments.

Nun beginnt das eigentliche Experiment. Hier hat das Taxi nur eine begrenzte (aber recht lange) Zeit um die Passagiere zur Fahne zu bringen. Danach wird es an eine zufällige Position zurückgesetzt. Dabei verliert es auch alle Passagiere, die es eventuell schon eingesammelt hat. Wenn das Taxi die Fahne erreicht hat (auch ohne Passagiere) oder in ein Feuer fährt, wird es ebenfalls zurückgesetzt. Bei jedem Zurücksetzen wird auch die Anzahl verbliebener Aktionsvorschläge an das Taxi wieder auf 20 zurückgesetzt. Das Taxi vergisst dabei nicht, was es bisher gelernt hat. Sie können das Taxi auch selbst mit dem Knopf „Setze den aktuellen Zustand zurück“ zurücksetzen. Nachdem das Taxi 12 mal zurückgesetzt wurde ist ein Trainingsdurchlauf beendet. Wenn ein ganzer Trainingsdurchlauf beendet ist, vergisst das Taxi alles Erlernte. Sie können einen Trainingsdurchlauf auch selbst beenden, wenn Sie der Meinung sind, dass das Taxi das gewünschte Verhalten erlernt hat. Hierzu drücken Sie bitte den Knopf „Beende das Training“.

Bitte melden Sie sich jetzt erneut bei dem Betreuer des Experiments.

Was war Ihr Eindruck bei diesem Experiment:

Nun wird das gleiche Experiment wiederholt, wobei das Taxi diesmal die Schritte etwas schneller ausführt.

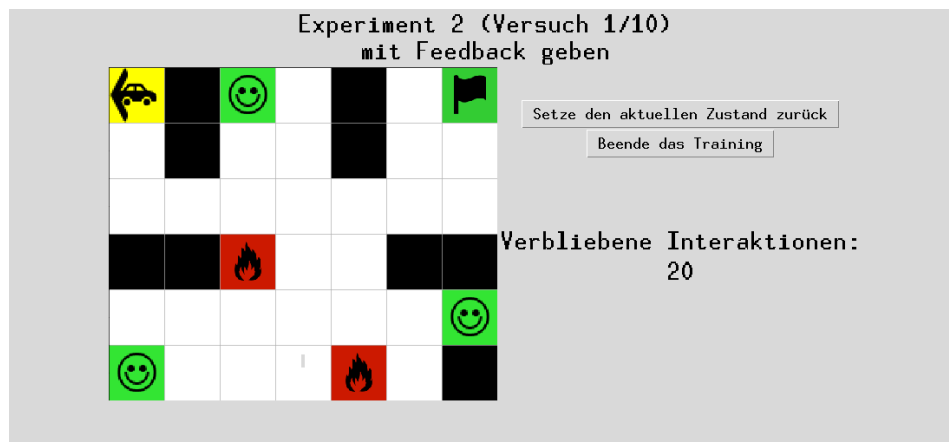
Was war Ihr Eindruck bei diesem Experiment:

Bitte lesen Sie sich die folgenden Instruktionen aufmerksam und vollständig durch, und melden Sie sich dann bei dem Betreuer des Experiments.

Aktionen bewerten

Das Ziel dieses Experiments ist, dass ein Taxi (gelbes Auto) möglichst schnell selbstständig zunächst alle Passagiere (grüne Smileys) einsammelt und diese dann zum Ziel (grüne Fahne) fährt. Dabei können alle weißen Felder verwendet werden. Die schwarzen Felder sind Mauern. Es schadet nicht gegen diese zu fahren, aber das Taxi kann sie nicht durchdringen. Darüber hinaus gibt es zwei Feuer, die das Taxi, wenn es in eines von diesen fährt, zerstören. Dies sollte vermieden werden.

Sie werden während des Experiments eine Darstellung, ähnlich der folgenden Abbildung, sehen:



Das Taxi kann sich zu jedem Zeitschritt um genau ein Feld nach rechts, links, oben oder unten bewegen. Mit Pfeilen um das Taxi wird angezeigt, welche Aktion es plant als nächstes auszuführen. Sie können mit der Tastatur bewerten, wie Sie die aktuell angezeigte Richtung für den nächsten Schritt einschätzen. Dabei stehen die Pfeiltaste nach oben und die Plus-taste für gut, während die Pfeiltaste nach unten sowie die Minus-taste für schlecht stehen. Sie können das Taxi nur 20 mal bewerten. Auf der rechten Seite wird Ihnen angezeigt, wie oft Sie das Verhalten des Taxis noch bewerten dürfen.

Sie können dies zunächst ausprobieren bis das Taxi die Fahne erreicht hat, in ein Feuer gefahren ist, oder Sie, wenn Sie alles verstanden haben, das Ausprobieren abbrechen.

Bitte melden Sie sich jetzt bei dem Betreuer des Experiments.

Nun beginnt das eigentliche Experiment. Hier hat das Taxi nur eine begrenzte (aber recht lange) Zeit um die Passagiere zur Fahne zu bringen. Danach wird es an eine zufällige Position zurückgesetzt. Dabei verliert es auch alle Passagiere, die es eventuell schon eingesammelt hat. Wenn das Taxi die Fahne erreicht hat (auch ohne Passagiere) oder in ein Feuer fährt, wird es ebenfalls zurückgesetzt. Bei jedem Zurücksetzen wird auch die Anzahl verbliebener Aktionsvorschläge an das Taxi wieder auf 20 zurückgesetzt. Das Taxi vergisst dabei nicht, was es bisher gelernt hat. Sie können das Taxi auch selbst mit dem Knopf „Setze den aktuellen Zustand zurück“ zurücksetzen. Nachdem das Taxi 12 mal zurückgesetzt wurde ist ein Trainingsdurchlauf beendet. Wenn ein ganzer Trainingsdurchlauf beendet ist, vergisst das Taxi alles Erlernte. Sie können einen Trainingsdurchlauf auch selbst beenden, wenn Sie der Meinung sind, dass das Taxi das gewünschte Verhalten erlernt hat. Hierzu drücken Sie bitte den Knopf „Beende das Training“.

Bitte melden Sie sich jetzt erneut bei dem Betreuer des Experiments.

Was war Ihr Eindruck bei diesem Experiment:

Nun wird das gleiche Experiment wiederholt, wobei das Taxi diesmal die Schritte etwas schneller ausführt.

Was war Ihr Eindruck bei diesem Experiment:

Abschließende Fragen

Welche Art der Information war für Sie einfacher zu geben, unabhängig vom Verhalten des Taxis:

- Es war einfacher Feedback zu geben
- Es war einfacher Aktionen vorzuschlagen
- Beides war gleich einfach (schwer)

Welche Methode hat nach ihrem Eindruck zu den besseren Ergebnissen geführt:

- Durch Feedback
- Durch vorzuschlagen von Aktionen
- Beides führte zu gleich guten (schlechten) Ergebnissen

Welche Geschwindigkeit des Taxis empfanden Sie als angenehmer um Feedback zu geben:

- Das schnelle Taxi
- Das langsame Taxi
- Beide waren gleich angenehm

Welche Geschwindigkeit des Taxis empfanden Sie als angenehmer um Aktionen vorzuschlagen:

- Das schnelle Taxi
- Das langsame Taxi
- Beide waren gleich angenehm

Sonstige Anmerkungen: