Ariel University

The Concept of Criticality in Artificial Intelligence

A doctoral thesis submitted in partial fulfillment of the requirements for the degree Doctor of Philosophy by

Yitzhak Spielberg

This work was prepared under the supervision of

Prof. Vadim Levit ,
Prof. Amos Azaria

March 2022

Acknowledgments

This thesis summarizes a wonderful journey of research. I would like to take this opportunity to thank all of those who made this journey possible, successful and fascinating.

Contents

Abstract

When people operate in the world or engage in a learning process they encounter different situations. There are situations where the action choice has only a minor influence on their lives, for example, when a person needs to choose between multiple dinner variants, and there are other situations where this influence is much larger (critical situations), for example, when a person needs to choose his life partner. Naturally, people adjust their physical and mental states to the type of situation they encounter and in accordance with common sense which says that critical situations require more intense thought and, in general, higher investment of resources. Clearly, also AI agents during training or upon deployment encounter states with different criticality levels. Thus, the central theme of this thesis is how AI agents can adjust their mode of learning/operation to the criticality levels of the states they visit.

In this thesis, I introduce a novel concept in Artificial Intelligence—the concept of criticality. Since criticality is a rather abstract notion, instead of providing a definition that holds for all AI domains, this thesis suggests different definitions of criticality for different AI domains. In addition to the introducing and discussing how the novel concept can manifest itself in a few chosen AI domains, the major focus of the thesis is to show that it can be utilized to help AI agents to learn more quickly and to operate more safely and efficiently. For this purpose the thesis presents 3 applications of criticality: Two in the context of Reinforcement Learning (chapters 2 and 3) and one in the context of AI Safety (chapter 4).

Whereas the first 4 chapters discuss applications of the novel concept in multiple domains of AI, chapter 5 does not deal with criticality in the narrow sense of the term but revolves

around a metric that can be regarded as closely related to criticality, namely, task difficulty. In the context of AI in education, the main question of that chapter is whether revelation of the task difficulty improves the student's learning experience. Task difficulty is related to criticality because it calls for an adjustment of the mode of operation: a difficult task requires the student to exhibit maximal mental and intellectual effort. Therefore, in the context of education, task difficulty can be considered as an aspect of criticality.

Chapter 1

Introduction

1.1 The notion of state criticality

Our decisions are not uniform in relation to the consequences they produce. Some of our decisions define to a large extent how our lives are going to look like for a long period of time. Examples of decisions of this category could be the choice of the life partner, the decision of whether to have children or not or, the choice of the profession. Many decisions that we make do not possess such fundamental importance as the examples above but still have a relatively tangible influence on our lives – at least for a certain period of time. An example for this category of decisions might be the choice of an apartment within an already chosen area (district) – in the case that a person (let's call him Bret) is planning to rent an apartment (rather than to buy one). On the one hand, the apartment possesses certain features that certainly will have at least a certain impact on Bret's life: the floor on which it is located, the noise level, the number of rooms, the distance from various facilities (supermarket, school, workplace, etc.). On the other hand, it is obvious that all of these influences are relatively minor in relation to the impact of the example mentioned above and moreover, Bret will always be able to move to another apartment if at some point he won't be happy with his current one. Whereas the decisions of the first and even the second type are relatively rare, most of the decisions that we need to make on a day-to-day basis have even less significant consequences. Examples of such minor decisions might be the choice of a movie or the choice of a dinner menu.

The term that we will use to denote the level of influence of a decision on the life of a person (in the context of AI - on the outcome/total reward) is the criticality of a decision (or situation/state/action choice). Decisions that have a minor influence on our lives have a relatively small criticality whereas decisions that impact our lives in a more significant manner have much higher criticality. We presented the examples above to illustrate the common sense understanding that in our lives we face decisions of different criticality levels.

The recognition that different decisions have different criticality levels is useful not only in teacher/student situations but also when people learn or operate by themselves. Usually, people adjust their mode of operation to the criticality level of the decisions they need to make, without thinking about criticality explicitly. To use the example presented above, if Bret is in a situation where he needs to decide whether to marry Heather or not, he will certainly not take this decision lightly. It is rather probable that he will take for himself much more time to ponder this question than he spends on day-to-day decisions. It is also rather likely that he will consider consulting on this issue with some of his friends and mentors – which is another behavior that he would not exercise for most of the less critical decisions he faces. Criticality-induced behavior adjustment can be observed not only in situations of major decision but also in situations that require motoric and mental skills such as various driving scenarios. It can be observed that drivers automatically slow down and increase their vigilance when they face a challenging situation on the road, such as black ice, heavy fog, or a complex junction.

The examples presented above show that humans adjust their mode of learning or operation to the criticality levels of the decisions they face and often benefit from this adjustment (otherwise they would consider such an adjustment). Likewise, also AI agents learn and operate via sequential decision making. Yet, unlike humans, most AI systems do not adjust their mode of learning/operation to the criticality of the environment state (decision). The main question that motivated the current piece of research is the question whether AI agents can benefit from adjusting their behavior to the criticality of the environment state. For this purpose, we explored different ways how AI agents can modify their learning and operational behavior in accordance with state criticality and whether these adjustments have the potential to speed up the learning procedure and/or to improve the agent's safety and efficiency upon deployment. To investigate this question, we present 3 ways of adjusting learning/operational behavior to the criticality of the states that the agent encounters or the actions he performs: 2 approaches in the domain of reinforcement learning and 1 approach in the domain of AI safety.

1.2 Skill-dependent criticality

In the previous section, criticality was introduced as a metric of a decision (or environment state) without taking into account any characteristics of the agent himself. Indeed, in many scenarios, this one-dimensional concept of criticality might be sufficient. For instance, the choice of a life partner is always a critical decision – independent of any characteristics of the person who needs to make this decision. Nonetheless, in many other scenarios, it might be important to consider not only the situation but also the agent's level of expertise. An example of such a scenario is a scenario in which a person needs to decide whether he is going to cross the street or to remain on the same side of the street. If the person is an adult who has high confidence in street crossing then, clearly, this situation has very low criticality. But in the case that the person is a little child who did not cross many streets until then, that same decision could be highly critical. Another example for skill-dependent criticality that is less extreme, could be a challenging driving scenario such as driving in heavy rain for a novice driver versus the same situation for an experienced driver.

In many scenarios, to measure skill-dependent criticality it is sufficient to consider solely the expertise of the decision-making agent. Yet, in other scenarios, the criticality level of a situation might depend on the skill levels of multiple agents. To illustrate why sometimes this is the case we might consider an example from the domain of sports: a situation in which a football player is about to take a free-kick from a promising position (a position from which there is a realistic chance to score a goal). The criticality of such a scenario depends on the expertise of 2 agents: the free-kick taker and the goalkeeper. In the case that the free-kick taker is a novice and the goalkeeper is a professional the criticality of the situation will be rather low because there is only a very tiny probability that a goal will be scored. But, if both the free-kick taker and the goalkeeper are highly skilled the situation can be regarded as rather critical since there is a much higher chance that it will result in a goal.

Criticality, as introduced, above is a metric that lives in the space of sequential decision-making tasks. Therefore the most natural application of criticality is in those sub-domains in AI that deal with active agents, such as reinforcement learning (RL), rather than those sub-domains in which there are no agents (such as classification of images or videos, segmentation of images, tracking objects in videos, etc.). Therefore, in this thesis, we will present two applications of criticality in the field of reinforcement learning: the varying stepnumber algorithm, which is related to n-step learning algorithms, and the criticality-based advice algorithm.

As mentioned above, the first application of criticality to RL (chp. 2) is connected to n-step algorithms. When using n-step algorithms to solve an MDP, the major challenge is the proper choice of n (the stepnumber), because this parameter is subject to the bias-variance tradeoff in the Q-function update. A mall n entails a small variance but a larger bias whereas a large n leads to a large variance but a small bias in the Q-function update term [1].The criticality-based varying-stepnumber algorithm (CVS) eliminates this problem by automating the choice of the stepnumber. In contrast to existing n-step algorithms, in CVS, n is not fixed over the entire learning procedure but is being determined by the algorithm itself in each learning step in accordance with the local criticality level of the environment. The central idea of the algorithm is that in low criticality domains of the state space the variance of the return is relatively small which enables the agent to apply a large stepnumber without paying the price of increased variance, while, in high criticality domains where the variance of the return is higher, the algorithm applies a small stepnumber, thereby reducing the variance.

The second application of criticality in RL (chp. 3) lives in the domain of advice-based RL algorithms. Since human advice is expensive, one of the major challenges that these algorithms

[1]https://www.endtoend.ai/blog/bias-variance-tradeoff-in-reinforcement-learning/

are aiming to solve is the proper selection of advice states (states, in which the AI agent is asking for advice). Many of these algorithms are using properties of the Q-function, such as variance with respect to the actions in a given state (e.g. in uncertainty-based advice). The drawback of these approaches is that especially in the early stages of the learning procedure, the estimate of the Q-function is very inaccurate, which leads to improper selection of advice states: states that do not require advice are selected as advice states, whereas states in which the agent would benefit from advice are not being selected. To make a step towards fixing this downside, we propose a criticality-based approach to the selection of advice states: criticality-based advice (CBA). In CBA, advice state selection is based on state criticality such that the probability of a given state to be selected for advice is proportional to its criticality. In chapter 3, we present 2 versions of criticality-based advice: p-CBA in which criticality is the only criterion, and m-CBA in which criticality is used in addition to the advice state selection criterion of a given advice selection algorithm.

1.3 Applying criticality to AI safety

While AI agents that live in artificial environments such as games are not able to cause any harm, AI agents that live in the real world can cause real damage to objects or people. As AI agents become more potent and are increasingly being deployed in safety-critical real-world domains, questions related to AI safety become more relevant. One of the central problems in the AI safety domain is to recognize unsafe states and unsafe actions. Unsafe states are defined by the property that an agent that is located in this state has a sufficiently high chance of causing harm to himself or to the environment. Hence, unsafe states can certainly be regarded be as critical states. Unsafe actions are actions that carry a sufficiently high risk of causing damage and thus can be viewed as critical actions. Therefore, the concept of state criticality might be particularly useful in the domain of AI safety.

The context in which we apply the concept of criticality to the AI safety domain is a setting where an AI agent (e.g. a robot) is trying to accomplish some high-level goal, such as preparing dinner, by successively formulating and executing action instructions (e.g. “add some salt to the soup”). Ideally, the AI agent should make sure that the instructions he is giving to himself do not cause any harm, but, since AI agents are not perfect there is always a risk that he will come up with an instruction that will put himself or his environment at risk. Therefore, there is a need for a safety agent (possibly a person) who will examine the AI agent’s action instructions. Although a safety agent would significantly reduce the AI risk, the downside of this very straightforward approach is obvious. In the case that the AI agent formulates instructions frequently the safety agent’s workload would be rather high. In chapter 4, we propose a criticality-based approach to reduce the safety agent’s workload. The strategy that we propose uses a filter that distinguishes potentially harmful instructions from instructions that are very likely to be safe. Thus, instead of being required to examine every instruction, the safety agent would need to check only those instructions selected by the filter. To build such a filter we use the metric of instruction criticality which is based on the linguistic analysis of the instruction.

1.4 Using criticality to prevent deskilling

In the previous sections, we have elaborated on the application of criticality in the reinforcement learning and AI safety domains and discussed ideas that leverage state/action criticality in order to make the agent's behavior safer and more efficient. Ultimately, one of the major motivations for the development of AI systems is making our lives easier and more comfortable. Many AI applications accomplish this purpose by freeing us from tasks that require a large amount of routine and only a minor investment of creativity. These applications can be considered as a subset within the more general category of automation devices. Examples from this category, which includes a rich collection of technologies, are numerous: calculators, autopilots in aircraft and self-driving cars, Google's auto-completion and automatic email response tools, medical software for doctors (which includes tools for diagnosis and treatment recommendation), intelligent IDEs for programmers which automatically produce code patterns and are equipped with auto-debugging tools, AI-based software for lawyers that can recommend trial strategies; software for business executives that automates decisions about hiring and pay.

Automation makes us more efficient and our lives more comfortable, but there is a price that we have to pay for it. Automation frees us from the effort of exercising our skills, but skills that are not being exercised degenerate. This is true both for such complex skills like driving a car, which are mostly motoric and routine-based, as well as for skills that require a high level of intelligence, such as the expertise needed for being a lawyer or an engineer. The consequences of deskilling are diverse. When a trader makes an unfortunate decision because he blindly trusts the software he is using, the damage he inflicts on the bank will be only monetary damage. However, when a pilot is not able to handle a routine flight situation in a situation where the autopilot system becomes dysfunctional, the outcome may be lethal. A study conducted between 2000-2010 indicates that a major portion of flight accidents in this period of time was caused by pilot errors that result from a lack of manual flight practice [16]. In recent years, the relevance of the deskilling phenomenon is increasing due to the enormous progress of AI technologies that successively replace humans in various domains and this leads to an increased need for the development of techniques that can prevent the erosion of skills due to automation.

There exist very ways to avoid the erosion of skill. One possible approach for the prevention of deskilling consists of augmenting AI systems and other automation systems with a skill preservation module. The purpose of this module would be to switch the system in manual mode so that the user would be required to exercise his facilities. In a self-driving car, for instance, the module would switch the car's driving mode from self-driving to manual and thereby enable the driver to practice his driving skills. One of the central problems that a skill preservation module would need to solve is to distribute periods of manual operation efficiently. Returning to the self-driving car example, efficient distribution of manual driving periods should ensure that the driver will be exposed to a rich collection of driving scenarios which should include different weather conditions, different times (day, night), different environments (urban, rural, highway) and different traffic conditions. Moreover, for efficient skill preservation, it might be important to expose the driver to critical driving scenarios, such as black ice, fog, and strong rain. For this purpose, the skill preservation module might need to make use of a criticality module that would evaluate the criticality level of various driving scenarios. The skill preservation module might then use these criticality estimates to select when to switch the car into manual mode.

1.5 Using criticality in education

While the debate regarding how much screen time is appropriate for children rages on among educators, psychologists, and parents, it is another emerging technology in the form of artificial intelligence and machine learning that is beginning to alter education tools and institutions and changing what the future might look like in education. It is expected that artificial intelligence in education will grow significantly in the coming years [2]. Even though most experts believe the critical presence of teachers is irreplaceable, there will be many changes to a teacher's job and to educational best practices. Therefore it might be particularly interesting to investigate the application of state criticality in AI systems for education.

Let us consider a learning scenario where the human student, for instance, a student pilot who is training in a flight simulator, is being assisted by an AI learning assistant. There are many ways how such an assistant might help the student to learn more efficiently. In the context of applications of criticality, one of the ways the agent might support the student in his learning process is by indicating to him, which situations are critical. A student who has a small amount of experience might easily misjudge the true difficulty of a flight situation. For instance, during the landing procedure, he might underestimate the various challenges induced by a particular weather condition. In such a scenario it might be particularly useful to indicate to the student, that the situation is indeed challenging. Upon being notified about the AI system's criticality estimate, the student will be motivated to focus all of his resources

[2] https://en.unesco.org/artificial-intelligence/education

and increase his vigilance in order to master the challenging situation. Consequentially, he will be more likely to succeed, and thereby, to learn more quickly.

In order to notify the student about highly critical states, the AI learning assistant needs to know how to distinguish them from less critical states. For that purpose, the learning assistant needs to have access to a criticality function: a function that assigns a criticality level to each environment state. One way to obtain a criticality function is to learn it from a human expert. In this case, the human expert (or a crowd of experts) should provide a training set of state/criticality tuples. This training set would then be used to train a criticality model, such as a neural network. Beyond enabling the AI learning assistant to model the criticality function, the procedure of collecting a training set of critical situations might also boost the performance of the human expert. It is well known, that in monotonous tasks (such as surveillance- and monitoring tasks), the operator's vigilance rapidly decays, but an operator that is being asked to record all critical situations will be much more likely to maintain his vigilance on a high level throughout the complete duration of the task [23].

It is certainly possible to imagine many more applications of state- or action criticality in AI, human education, and many other domains. Yet, within the limits of this thesis, we can discuss only a few selected applications of this novel concept. As mentioned previously, 2 of the 4 applications of criticality in AI that we present in this thesis are related to the domain of reinforcement learning. Therefore, the first chapter of the thesis will provide a compact introduction to the field of RL.

1.6 Publications

Some of the results that appear in this thesis were published in:

1. Y.Spielberg, A.Azaria , The Concept of Criticality in Reinforcement Learning , in Proceedings of ICTAI 2019 [61]. This paper is the basis for chapter 3.

2. Y.Spielberg, A.Azaria, The Concept of Criticality in AI Safety , arxiv 2020 [63]. This paper is the basis for chapter 5.

3. Y.Spielberg, A.Azaria, Criticality-based Varying Step-number Algorithm for Reinforcement Learning , IJAIT 2021 [64]. This paper is the basis for chapter 3.

4. Y.Spielberg, A.Azaria, Revelation of Task Difficulty in Al-aided Education, in Proceedings of ICTAI 2021 [65]. This paper is the basis for chapter 6.

5. Y.Spielberg, A.Azaria, Criticality-Based Advice in Reinforcement Learning, in Proceedings of AAAI 2022 (student abstract), also under review for COGSCI 2022.[66]. This paper is the basis for chapter 4.

Chapter 2

Preliminaries: Reinforcement Learning

2.1 What is Reinforcement Learning ?

Since 2 applications of criticality that are being presented in this thesis live in the domain of reinforcement learning (RL), in this chapter I will provide a short introduction to this field. Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. It differs from supervised learning in that labelled input/output pairs need not be presented, and suboptimal actions need not be explicitly corrected. Instead the focus is finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).The environment is typically formulated as a Markov decision process (MDP), as many reinforcement learning algorithms for this context utilize dynamic programming techniques. An MDP constitutes

- a state space S
- an action space A
- the transition probability measure $P_a(s, s')$
- the reward function $R_a(s, s')$

Policy and state-value Function

The agent's action selection is modeled as a map called policy:

$$\pi : A \times S \to [0,1] \pi(a, s) = \Pr(a_t = a \mid s_t = s)$$

The policy map gives the probability of taking action a when the agent is in state s. The state-value function $V_\pi(s)$ is defined as the expected return starting with state s, and successively following policy π . Hence, roughly speaking, the state-value function estimates how good it is to be in a given state. It is defined by:

$$V_\pi(s) = E\left[R \mid s_0 = s, \, \pi\right]$$

where the random variable R denotes the return, and is defined as the sum of future discounted rewards:

$$R = \sum_{t=0}^{\infty} \gamma^t r_t$$

Here r_t is the reward at step t , and $\gamma \in [0, 1]$ is the discount-rate.

The learning algorithm must find a policy with maximum expected return. From the theory of MDPs it is known that, without loss of generality, the search can be restricted to the set of so-called stationary policies. A policy is stationary if the action-distribution returned by it depends only on the last state visited (from the observation agent's history). The search can be further restricted to deterministic stationary policies. A deterministic stationary policy deterministically selects actions based on the current state. Since any such policy can be identified with a mapping from the set of states to the set of actions, these policies can be identified with such mappings with no loss of generality.

2.2 The action-value function

Value function approaches attempt to find a policy that maximizes the return - the optimal policy. A policy is called optimal if it achieves the best expected return from any initial state. An optimal policy π^* is defined by:

$$V_{\pi^*}(s) = V^*(s)$$

where $V^*(s)$ is the optimal state-value function:

$$V^*(s) = max_\pi V_\pi(s)$$

Although the state-value function suffices to define optimality, the learning algorithm uses a function, that depends not only on the state, but also on the action - the action-value function (also called Q-function). Given a state s, an action a and a policy π , the action-value of the pair (s, a) is defined by:

$$Q_\pi(s, a) = E[R \mid s, a, \pi],$$

2.3 Some learning algorithms

This section presents some of the RL algorithms that will be mentioned in this thesis.

2.3.1 Q-Learning and SARSA

Q-learning and SARSA are 2 of the simplest model-free reinforcement learning algorithms to learn the value of an action in a particular state. These algorithms do not require a model of the environment (hence ”model-free”), and it can handle problems with stochastic transitions and rewards without requiring adaptations. For any finite MDP, Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state. Q-learning and SARSA can identify an optimal action-selection policy for any given MDP, given infinite exploration time and a partly-random policy. The difference between these algorithms is located in the update targets for the Q-function. The update rule for Q-learning is given by:

$$Q(S_t, A_t) := Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$$

and the update rule for SARSA is:

$$Q(S_t, A_t) := Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

2.3.2 n-step learning methods, Q (λ) and Monte-Carlo

With one-step learning methods the same time step determines how often the action can be changed and the time interval over which bootstrapping is done. In many applications one wants to be able to update the action very fast to take into account anything that has changed, but bootstrapping works best if it is over a length of time in which a significant and recognizable state change has occurred. With one-step learning methods, these time intervals are the same, and so a compromise must be made. n-step methods enable bootstrapping to occur over multiple steps, freeing us from the tyranny of the single time step. Any one-step learning algorithm can be transformed into an n-step algorithm by moving the update target n-steps ahead. In n-step SARSA, for example, the update rule for the Q-function is given by:

$$Q(S_t, A_t) := Q(S_t, A_t) + \alpha(\sum_{k=1}^{n} \gamma^{k-1} R_{t+k} + \gamma^n Q(S_{t+n}, A_{t+n}) - Q(S_t, A_t))$$

Another RL algorithm that is appears in this thesis is Watkins's $Q(\lambda)$. This algorithm belongs to the family of n-step learning algorithms but instead of the n-step return (as in ordinary n-step algorithms), $Q(\lambda)$ uses a linear combination of n-step returns. The interested reader can find the algorithm in [68] (chp. 12).

Another algorithm that will be used to as a baseline for criticality-based learning algorithms is the Monte-Carlo algorithm. In this learning algorithm the state corresponding to the update target is the terminal state. Thus, the Monte-Carlo update rule is:

$$Q(S_t, A_t) := Q(S_t, A_t) + \alpha(\sum_{k=t}^{T-1} \gamma^{k-t} R_k + \gamma^{T-t} Q(S_T, A_T) - Q(S_t, A_t))$$

Chapter 3

The Concept of Criticality in Reinforcement Learning and it's Application to n-step Algorithms

3.1 Introduction

Our decisions are not uniform with relation to the consequences they produce. Some of them can be easily and immediately forgotten, while others have very significant consequences that may influence us for the rest of our lives. “What should one have for dinner?”, “Should one invest two extra hours to work on one's project or spend the evening watching a movie?”, “Which route should one take to work?” These decisions are almost meaningless, since they do not have any enduring influence on a person's life.

On the other end of this spectrum are questions such as: “In which country does one want to live?”, “Which profession should one possess?” , “How much to invest in health?”, and

“How to educate one’s children?”. These decisions might influence us personally and as well as our close ones for many years to come and therefore require profound consideration.

In reinforcement learning an autonomous agent is trained to act in a way that maximizes its expected return in a given environment. During the learning process the agent is situated in a certain state and is required to choose one particular action from a set of possible actions. Clearly, in some situations, different actions may lead to very similar expected return values, while in other situations, different actions may lead to very different expected returns. In the former case we may say that the situation (or state) that the agent is visiting is not very critical, as it does not matter that much which action the agent will choose. However, the second situation appears to be critical, as an agent failing to take an optimal action may result at a very low outcome.

In this chapter we introduce the concept of criticality. The criticality level of a state indicates how much the choice of the action influences the agent’s performance. The concept of criticality is inspired by the intuition that a state in which the choice of action matters should be considered as more critical, than a state in which it doesn’t.

We believe that the concept of criticality is particularly useful in the context of human-aided reinforcement learning, where the learning agent receives criticality information from a human trainer. In such a learning scenario there might be algorithms that use criticality in order to boost the agent’s performance. In this chapter we present one such learning algorithm: the criticality-based varying step-number algorithm (CVS). CVS might be regarded as an algorithm that is closely related to the class of n-step learning algorithms (with a fixed step-number), such as n-step SARSA and n-step Tree Backup, but with a flexible step-number. By using a flexible step-number, CVS does not suffer from the central problem of fixed step-number algorithms: the problem of choosing an appropriate step-number.

We compare the performance of CVS to other reinforcement learning methods in three different domains. While the first two domains are quite simple, they provide strong motivation for the use of CVS. The final domain is an Atari-based domain, namely the game of Pong. We show that CVS outperforms other baselines in these three domains.

3.2 Related Work

Reinforcement learning based methods have recently shown great success in many domains, including Atari games [46], Go [60], and autonomous vehicles [58, 56, 27, 28]. Human-aided reinforcement learning introduces methods that enable the reinforcement learning agent to take advantage of human knowledge in order to learn more efficiently. Prior work in this relatively new area of research has taken a variety of forms. In the first part of this section we present some of these approaches. In the second part we will focus on past research which is more closely related to criticality and to n-step algorithms.

One of the ways in which a reinforcement learning agent can profit from human knowledge is by reward-shaping: engineering an artificial reward function by synthesizing the human's understanding of the environment with the environment's reward function. Reward shaping techniques are particularly appropriate in sparse reward environments such as environments in which all states with the exception of a few terminal states have a zero reward. One of the pioneering reward shaping approaches [42] utilized the human's intrinsic knowledge of the environment. An alternative reward shaping algorithm is the TAMER framework [34] and (the related Deep TAMER [77] for high-dimensional state spaces) which fits a parametric model of the human reward function using human feedback provided during the interactive learning procedure.

Another viable class of methods involve learning from human demonstration [57, 36, 37]. The Human-Agent Transfer algorithm [70] is one example from this class. It combines transfer learning, learning from demonstration and reinforcement learning. Another interesting representative of this class synthesizes learning from demonstration and reward shaping [15].

Advice plays an important role in the context of human-agent interaction. Advice may be provided by the agent to the human (e.g. [9, 8, 7, 52]) or be provided by the human to assist the agent in its learning process. Indeed, advice-based techniques are also used in human-aided reinforcement learning. In contrast to reward shaping approaches, these techniques instruct the agent directly by feeding it with human advice. Advice-providing methods can be applied in both value-function based and policy-gradient based learning algorithms [25, 33].

We now consider work that is more closely related to criticality. We defined the criticality of a state as a subjective measure of the Q-function's variability with respect to the actions. In our literature research we wanted to know whether somewhat similar concepts have been proposed previously. Since similar concepts can be formulated in many different ways the literature research was rather challenging. We found only one concept which is closely connected to criticality and we can not guarantee that we did not miss any other relevant ideas. This concept, called "Importance", was introduced by [74]. The importance of a state is defined by:

$$I(s) = \max_a Q(s,a) - \min_a Q(s,a)$$

This chapter proposes multiple algorithms that determine in which states the agent would ask the human teacher for advice and importance was one of the measures which was utilized for this purpose. The ideas formulated in that paper were extended by [4] who suggested that advice should be initiated by both the teacher and the agent. The concept of importance is certainly similar to criticality, since it also measures the Q-function's sensitivity with respect to the action choice. However, there are also two significant differences between these two concepts. First, the importance of a given state is defined by the current estimate of the Q-function and therefore will change in the course of the learning, while the criticality of a state will not. Second, in contrast to importance, criticality is a purely subjective estimate, which reflects the teachers view of the environment.

After having discussed work that is related to the concept of criticality we mention some of the prior research on a topic that is a central problem in n-step algorithms (since CVS is closely related to n-step algorithms): The bias-variance trade-off in n-step algorithms. All n-step algorithms relate to the bias-variance trade-off, since the update of the Q-function suffers from a large bias if the value of n is small, and from large variance if n is big. Various techniques have been developed to tackle this challenge.

De Asis [6] addresses this problem for off-policy n-step TD methods, such as n-step Expected SARSA, via the introduction of so called *control variates*. These special terms have the impact of an expectation correction. Therefore they can be used to decrease the bias of the n-step return.

Jiang et al. [32] propose an alternative solution for this problem for the prediction task (not the optimal control task). They introduce an unbiased estimator, which corrects the current estimate of the value function $\hat{V}(S_t)$. This estimator is robust in the sense that it remains unbiased even when the function class for the value function is inappropriate.

Richard Sutton et al. [69] suggest an improvement of TD(λ) that achieves an effective bias reduction for the updates. This beneficial effect is a consequence of specific weights that are being assigned to any given update of the value function. The proposed variant of TD(λ) is particularly useful for off-policy learning, where ordinary TD(λ) suffers from a deficit of stability.

Unlike all of the above mentioned approaches our method does not manipulate the updates of the (action) value function a-posteriori; instead, it chooses the appropriate step-number for the update a-priori. This is done by using the criticality function, which is closely related to the update's variance. Therefore, in a broad sense, we can regard the CVS algorithm as a technique that speeds up the learning by controlling the variance of the updates.

3.3 The Concept of Criticality in Reinforcement Learning

3.3.1 A Definition of Criticality

In the context of reinforcement learning the criticality of a state indicates how much the choice of action in that particular state influences the expected return. We define the criticality of a state as a measure of variability of the expected return with respect to the available actions. The criticality is a value in the range of [0,1], where 0 represents no variability between the expected return of the actions (for example, if there is only a single action, or if all actions result in the same expected return), and 1 represents high variability between the expected return of the actions (for example when some actions result in a very high expected return, while other actions result in a very low expected return). Variability is related to variance, such that a variance of 0 in the expected return entails variability of 0 (and thus criticality of 0); while a variance greater than 0 entails criticality greater than 0.

The recognition that some states are more critical than others is particularly useful in learning situations that include a teacher and a student. An example of such a learning situation is a driving lesson. If a student driver approaches an obstacle on the road, her teacher may state to her that she must watch out, without suggesting exactly which action to take (e.g. slowing down, turning the wheel right or left etc.). This warning will motivate the student to pay more attention to the situation and therefore it will be more likely that she will be able to avoid the obstacle. Moreover, even if the car later hits that obstacle, the student will understand that she probably took a wrong action back when the teacher warned her, and this understanding will help her to learn more efficiently. The situation of a driving lesson possesses the characteristics of a human-aided reinforcement learning scenario. The learning agent finds himself in a certain state and needs to choose one action from an array of possible actions. The human teacher informs him about the criticality level of the current state. The learning agent then utilizes the criticality information in order to improve his learning strategy (for example by implementing the CVS algorithm, which will be presented in this chapter).

We introduced criticality in a way that portraits it as a human centered concept, in the sense that it is a *person's* estimate of the spread of consequences with respect to the available actions. Therefore, the definition implies that the criticality function (that is, the function that assigns a criticality level to each state of the environment) of a given environment is not unique, but can be any element from a whole class of functions that are loosely defined by the variance of the expected return. Beyond this type of diversity there is another dimension of freedom in the concept of criticality, which comes from the absence of the optimal policy in its definition. Since in many environments a human does not exactly know the optimal policy, any definition of criticality that includes the optimal policy in an explicit manner (for example the variance of the optimal Q-function in a given state with respect to the actions) would not be human-friendly.

3.3.2 Obtaining criticality from a model or from the environment

So far we have discussed a scenario in which the human trainer provides the criticality level in every state encountered by the learning agent. If the human can implement the criticality measure in a functional form (as we later use in the experiment sections), the workload on the human trainer is reasonable. However, a setting, where the human trainer provides criticality in real time during the learning procedure, might be unfeasible for two reasons. Firstly, a learning procedure that takes long would require a substantial investment of time from the trainer. Secondly, because the effort required for the estimation of the criticality level of one single state accumulates over the complete learning session, the trainer might be exposed to a tremendous workload.

There are multiple approaches towards a solution for this problem. The first one involves the human trainer and a criticality model. In this approach the trainer is being asked to give his criticality estimates on a set of states. On the basis of this set a criticality model for the given environment is learned. During the reinforcement learning process, the agent obtains its criticality input from the criticality model. An alternative approach is for the reinforcement learner to obtain the criticality level from the environment directly, without the necessity of a human trainer. Since, according to the definition, criticality is related to the variance of the action-value function with respect to the actions, this variance (possibly normalized, because the criticality needs to be in [0,1]) can be used as an estimate of the criticality.

3.3.3 Policy-dependent Criticality

It may not always be obvious which states should be considered critical and which states should be considered as non-critical. For example, a car driving on a straight road with no traffic may seem as being in a non-critical state. However, a driver that suddenly turns the wheel right (or left), may result in hitting a wall, and action that is likely associated with a negative reward. This could imply that the state was in fact a critical state. However, in this example the variance might be low, since most actions such as changing the speed or modestly turning the wheel won't have any meaningful impact. Therefore, in certain learning situations, it might be necessary to refine the definition of criticality in order to capture these scenarios.

One option is to multiply each expected return by the probability that the agent will take each action, and then compute the weighted variance (rather than the plain variance). This definition may be closer to what humans view as critical states. It would require transforming the weighted variance to a value between 0 and 1 by some kind of normalization procedure. According to this more sophisticated definition, the criticality is no longer associated only with a state, but is now associated with a policy as well, and may therefore change over time. This is intuitive, since when the agent plays better, different states may seem more critical. For example, for a novice basket-ball player, a position from which a 3-point opportunity exists, seems less critical (because the player is very likely to miss) than for a professional player, who is more likely to score.

3.4 The Construction of Criticality Measures in Various Environments

So far we have defined the concept of criticality in reinforcement learning and we have discussed how it can be refined and expanded in order to guarantee more robustness in various learning situations. We have stated that a central feature of the concept of criticality, the way we envision it, is its human-friendliness.

Therefore we formulated our definition in a manner that leaves multiple degrees of freedom by linking criticality only loosely to both the optimal policy and the variance of the Q-function in a given state. In this section we want to convey to the reader an intuition of the way a criticality measure can be constructed by presenting plausible criticality measures in multiple environments.

The Atari Pong environment consists of two rackets, of which one is the agent and the other is the opponent, a ball, and a playing field.

The agent receives a reward of +1 when he scores a point, and a reward of -1 when the opponent does. The Pong game has an interesting characteristic: when the ball is moving away from the agent, its actions are irrelevant. Plausible criticality measures can be constructed on the basis of this characteristic. The simplest criticality measure could assign a criticality of zero to each state, in which the ball moves away from the agent and a maximal criticality of 1 to each state, in which the ball moves towards the agent. A slightly more sophisticated criticality measure might use some decreasing function of the distance between the ball and the agent in those states, where the ball is moving towards the agent, since the agent's actions become more critical, as the ball is coming closer to it.

Pacman is a classic game, which involves the titular character in an enclosed maze filled with individual dots, or pellets. The goal is to consume all of the pellets while avoiding four ghosts that wander around the maze. If a ghost touches the agent, it loses a life, which can be regained at certain point values. The maze also contains four large "power pellets", which give the agent temporary invulnerability, allowing it to consume the ghosts and earn additional points. Throughout the game, fruits appear in the center of the maze, which can be consumed for earning additional points as well. There are various situations in the game that might be considered as critical. It is very important for the agent to avoid an encounter with a ghost. Therefore, a state in which the agent is close to a ghost, might be viewed as critical. Another type of critical state might be a state in which the agent is close to a fruit or a power pellet, since these items are beneficial to it. Another critical situation might be a state in which the agent is close to a pellet, and there are only a few pellets left in the field. Similarly to Atari Pong the criticality function in all these states might be defined as some decreasing function of the agent's distance to the relevant object (ghost/fruit/pellet).

For another example of criticality, consider a life-guard agent, which is required to ensure the safety of people bathing in a pool. The agent may perform several actions such as throwing a life ring to different locations at the pool. Clearly, one of the most important tasks for such an agent is to detect which states are critical and which states are not. A critical state would be a state in which a person is having some difficulty to remain above water. Taking no action or throwing a life ring to an incorrect location when a person is drowning, may have catastrophic consequences. However, throwing a life ring to any location, when there is no person requiring help, is likely to result in a very minor penalty.

Self-driving cars are currently one of the most attention-grabbing applications of artificial intelligence. Since reinforcement learning techniques are instrumental in teaching them to drive autonomously, it might be particularly interesting to discuss the construction of a criticality measure which might make the learning procedure more effective. Obviously, the list of critical traffic situations might become very long, because of the complexity of real-world scenarios, so we will limit our scope and indicate only three major categories of critical states. The first type of critical situations is related to weather conditions. It might include scenarios such as black ice and dense fog. Another category of critical states is related to the complexity of the situation. This category might include such situations as left turns, complex junctions and moments in which the behaviour of nearby vehicles is unclear. The third type of critical situation is related to the traffic density. This category might include areas that are highly populated by pedestrians or playing children.

3.5 CVS

In this section we introduce a practical application of criticality in reinforcement learning: the criticality-based varying step-number algorithm (CVS) - a flexible step-number algorithm that utilizes criticality information, in order to avoid the problem of choosing an appropriate step-number in n-step algorithms (which use a fixed value of n), such as n-step SARSA and n-step Tree Backup [64].

3.5.1 The Relation between Criticality and the Step-number

All prominent n-step reinforcement learning algorithms, such as n-step SARSA, n-step Expected SARSA and n-step Tree Backup, use a fixed step-number n for bootstrapping, which stays constant both in the course of an episode and during the complete learning process. In our approach we use a varying step-number that is specific to each state encountered during an episode, and we use criticality to determine the appropriate step-number for a given state.

In order to develop some intuition on the way in which criticality could be used to determine an appropriate step-number, we present a simple example. In this example we will work with the n-step SARSA return:

$$G_{t:t+n} = R_t + \gamma R_{t+1} + ... + \gamma^{n-1} R_{t+n-1} + \gamma^n Q(S_{t+n}, A_{t+n})$$

Let us assume that in our environment most of the states have only one available action, and that there is no randomness in the Markov Decision Process (MDP), that is, a given state action pair determines the next state. Let us further assume that during the learning process the agent encounters some sequence of states-action pairs:

$$(S_0, A_0), (S_1, A_1), (S_2, A_2), (S_3, A_3), (S_4, A_4)$$

of which only S_3 has multiple actions available. In this situation, obviously S_0, S_1, S_2 should be assigned a criticality of 0 (since the agent has no choice, and therefore its action has no influence on the final return value, i.e. the variability of the return is 0) whereas for simplicity we will assign to S_3 a criticality of 1. Clearly, whenever the agent arrives at S_0, the next states it visits will always be (S_1, S_2, S_3). We would like to determine $n \in \{1, 2, 3, 4\}$ should be used for the n-step return $G_{0:n}$ that will serve as the update target for $Q(S_0, A_0)$.

Consider the simple 1-step SARSA. This algorithm will update $Q(S_0, A_0)$ towards $G_{0:1}$ and in the next step $Q(S_1, A_1)$ towards $G_{1:2}$. These updates will be repeated in each episode where these states are being visited, so it is easy to see that asymptotically $Q(S_0, A_0)$ will be updated towards $G_{0:2}$. Therefore, there is no benefit from selecting $G_{0:1}$ as the update target for $Q(S_0, A_0)$ versus selecting $G_{0:2}$. Moreover, the selection of $G_{0:2}$ may speed up the convergence. Using the same argument we can conclude that $G_{0:3}$ is a better update target than $G_{0:2}$. However, updating $Q(S_0, A_0)$ towards $G_{0:4}$ may not be the best choice, since the agent may choose a different action at S_3, which will lead it to a state that is different from S_4.

We now discuss the question of how to construct a criticality-based algorithm that would choose $n = 3$ for the update target $G_{0:n}$ for $Q(S_0, A_0)$. One way of doing so is by simply choosing the smallest $n > 1$ for which S_n has a criticality above a given threshold (e.g. 0.5). This algorithm looks appealing due to its simplicity and works well in our simple example. Yet, it has two downsides. First, it is not clear what the threshold should be. Second, it is invariant to the criticality of all the states that precede the S_n which corresponds to the chosen update target $G_{0:n}$ as long as they remain beneath the threshold. This is an important point in a situation where the individual states in a certain domain have a criticality beneath the threshold but the domain of the state space as a whole has a high *cumulative criticality*; that is: the sum of the criticality over states that belong to this domain is high. These considerations motivate an alternative way to use criticality for the choice of a good update target: The CVS algorithm, which we present in the next section.

3.5.2 The CVS Algorithm

We now present a method that on the one hand will choose the appropriate update target S_3 in the example from the previous section, and on the other hand will avoid the two downsides of the threshold criticality approach. This method uses the idea of cumulative criticality; It chooses the Q-value of the state S_n with the lowest number n for which $crit(S_1) + crit(S_2) +$

$... + crit(S_n) \geq 1$ as the update target. The choice of the value 1.0 as the treshold for the cumulative criticality can be motivated if we consider a binary criticality function that assigns a value of either zero or one to a given state. In that case it would be desirable that the Q-values of the critical states (those, whose criticality is one) would be used as update targets. This method does not suffer from any of the disadvantages of the first method: there is no necessity to determine a threshold and it will produce small step-numbers in more critical domains of the state space. We name this algorithm “Criticality-based Varying Step-number” (CVS)(alg. 1). The update target also depends on the specific algorithm to which CVS is applied: E.g. in the CVS version of Q-Learning it will be $\max_a Q(S_n, a)$; in the CVS version of SARSA it will be $Q(S_n, A_n)$ etc.

Algorithm 1 The CVS algorithm (SARSA version)

```
Given: criticality function Crit()
  $CritCum(s, a) = 0$ for all s,a (cumulative criticality)
  WaitList= {} (states waiting for update)
  pick initial state $S = S_0$ and action $A = A_0$ greedily
  while $S \neq Terminal$
    add $(S, A)$ to WaitList
    observe $R, S'$
    pick $A'$ greedily
    for $(\hat{S}, \hat{A})$ in WaitList
      if $CritCum(\hat{S}, \hat{A}) \geq 1$:
        update $Q(\hat{S}, \hat{A})$ towards update target $Q(S', A')$
        delete $(\hat{S}, \hat{A})$ from WaitList
        $CrtCum(\hat{S}, \hat{A}) = 0$
      else:
          $CritCum(\hat{S}, \hat{A}) += Crit(S')$
    S,A = S',A'
  for $(\hat{S}, \hat{A})$ in WaitList
      update $Q(\hat{S}, \hat{A})$
      towards update target $Q(S', A')$
```

3.6 Evaluation of CVS in the Road-Tree environment

In this section we introduce the Road-Tree environment, an environment that is particularly appropriate to understand the benefits of CVS. We test the algorithm against a number of widely used reinforcement algorithms in order to prove it's efficiency. By default, if not specified otherwise, we do not discount the reward (i.e., $\gamma = 1$) and our initial Q-function is constant over the state-action space. Our default values for epsilon and the learning rate are $\epsilon = 0.1$, and $\alpha = 0.1$. Moreover, in all experiments that are mentioned in this chapter we use the Q-Learning version of CVS.

3.6.1 The Road-Tree environment

In order to test CVS, we construct a plain environment, named Road-Tree, which has a natural criticality function corresponding to it. Road-Tree has a tree-like structure. The agent starts at the root and always moves in one direction–downward. There are two types of states. In a simple state there is only one possible action. In a junction state the agent needs to choose between multiple roads. The reward upon stepping onto a simple state is always zero. The reward is nonzero only upon reaching a junction or a terminal state. Moreover the reward may vary across junctions and terminal states. Figure 3.1 illustrates a simple Road-Tree environment. The numbers in the junctions represent the rewards. The numbers on the edges show the distance between the two corresponding junctions, which is the number of simple states between them (a distance of n indicates $n - 1$ simple states).

The very natural criticality function that we are going to use in the Road-Tree environment assigns zero to a simple state and one to a junction or terminal state.

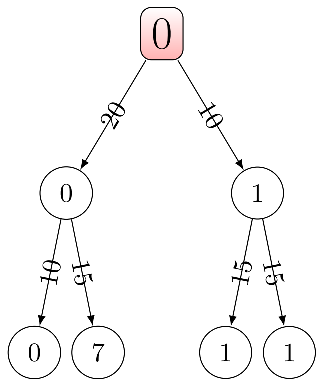


Figure 3.1: Road-Tree example. The number in a critical state (junction or terminal state) represents the reward in that state. The number on an edge is the distance between the corresponding nodes. The optimal policy is defined by initially going to the left and then to the right, ending up at the terminal state that has a reward of $r = 7$.

3.6.2 CVS vs. Q-Learning, Q(λ) and Monte Carlo

We now compare the performance of CVS against that of Q-Learning in the 2-level Road-Tree from fig. 3.1. Clearly, the optimal policy is defined by initially going to the left and then to the right, ending up at the terminal state that has a reward of $r = 7$. In Q-Learning, due to the relatively big distance between the intermediate junction that has a reward of $r = 0$ and the optimal terminal state, the optimal reward ($r = 7$) will be backpropagated to the intermediate junction very slowly. The other intermediate junction that has a reward of $r = 1$ will be much more attractive to the agent and therefore, the agent might remain in that nonoptimal path for a long period of time. Conversely, the CVS agent will backpropagate the optimal reward terminal state to the intermediate junction immediately after the first visit and therefore should quickly converge to the optimal policy. The plot on fig. 3.2 confirms our elaboration. The Q-Learning agent needs about 6000 episodes to converge to the optimal policy; the CVS agent, in contrast, converges after 1000 episodes.

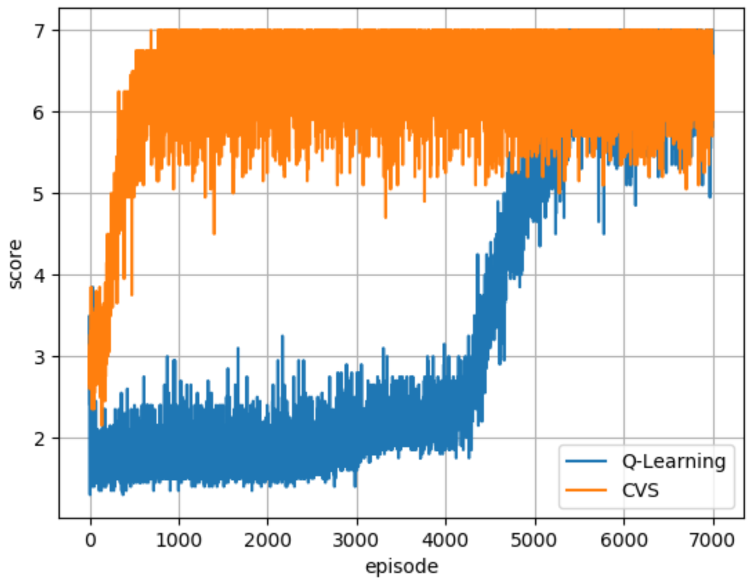


Figure 3.2: CVS against Q-Learning in a 2-level Road-Tree environment (Figure 3.1). Average scores over 20 runs. CVS converges to the optimal policy after about 1000 episodes, while Q-Learning requires about 6000 episodes.

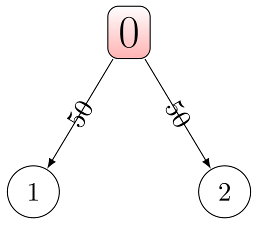


Figure 3.3: Plain Road-Tree Environment with only two roads. Since both roads have the same length it is not obvious that CVS will outperform Q(λ).

Next we test CVS against Watkin's Q(λ), which is one of the popular algorithms in reinforcement learning (we set $\lambda = 0.9$). We perform this evaluation in a very simple Road-Tree environment that contains only two roads that have the same length (see Figure 3.3). If the optimal road was much longer than the nonoptimal one it would be obvious that CVS would outperform Q(λ), because of considerations that are very similar to the Q-learning scenario. Plot 3.4 shows that even in this more challenging scenario CVS learns faster than Q(λ). It can be seen that Q(λ) struggles to make any progress; in contrast, CVS takes about 200 episodes to converge to the optimal policy.

In the previous example, in which CVS outperformed Q(λ), the algorithm functioned exactly the same way as Monte Carlo (MC) would, by choosing the Monte Carlo return as the update target for each of the previous states. This observation immediately raises the question, whether we can construct an example where CVS would outperform Monte Carlo.

Figure 3.5) presents a 3-level Road-Tree with two junctions on the second level and a multitude of terminal states. The terminal state of the optimal trajectory is hidden among 99 siblings, which all have a very bad reward. Since the first visit of the right branch will probably end up at one of these bad siblings, the negative return of the trajectory will be backpropagated to the root immediately and therefore a Monte Carlo agent will avoid the

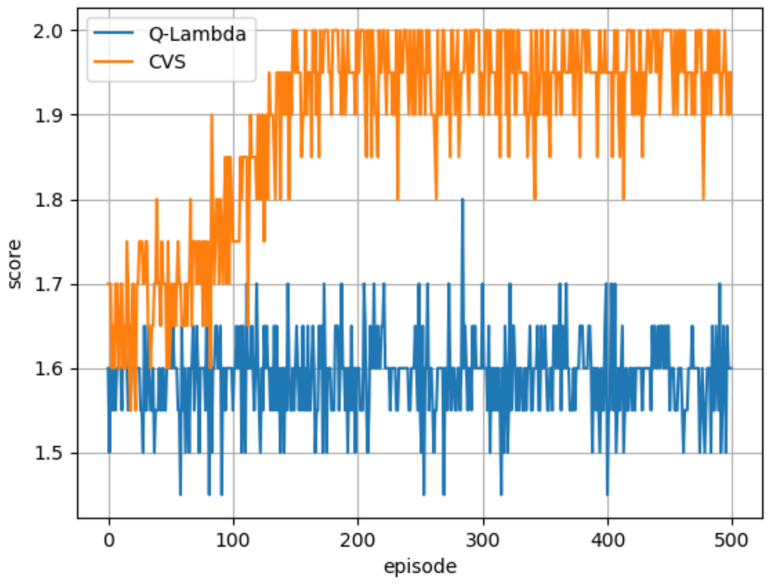


Figure 3.4: CVS performance against Q(λ) in the plain Road-Tree environment fig. 3.3. Average scores over 20 runs. Q(λ) struggles to make any progress; in contrast, CVS takes about 200 episodes to converge to the optimal policy.

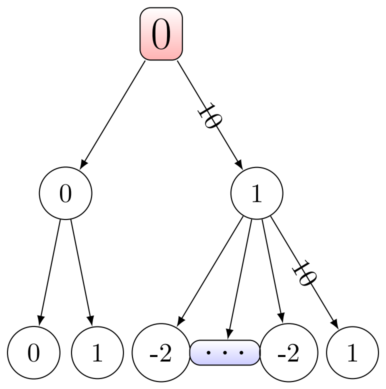


Figure 3.5: Road-Tree Environment with two levels and a large number of branches. The terminal state of the optimal trajectory is “hidden” among 19 suboptimal siblings.

right branch, which in fact is the optimal one. In contrast to Monte Carlo, the CVS agent will choose one of the n-step return one of the intermediate junctions,$G_{root:intermjunc}$, as the update target for the root and therefore will not lose its interest in the right branch so quickly. As a consequence, it is much more likely that it will require fewer episodes to discover the optimal trajectory. Indeed, the experiment confirms our intuition. From figure 3.6 we can imply that, as expected, the Monte Carlo agent visits the left junction most of the time and, as a consequence, fails to identify the optimal policy. We can also infer from the plot that, in contrast to the Monte Carlo agent, the CVS agent visits the right junction much more frequently. The plot shows that the optimal trajectory was visited for the first time after about 30 episodes and from there on the CVS agent stayed with it most of the time.

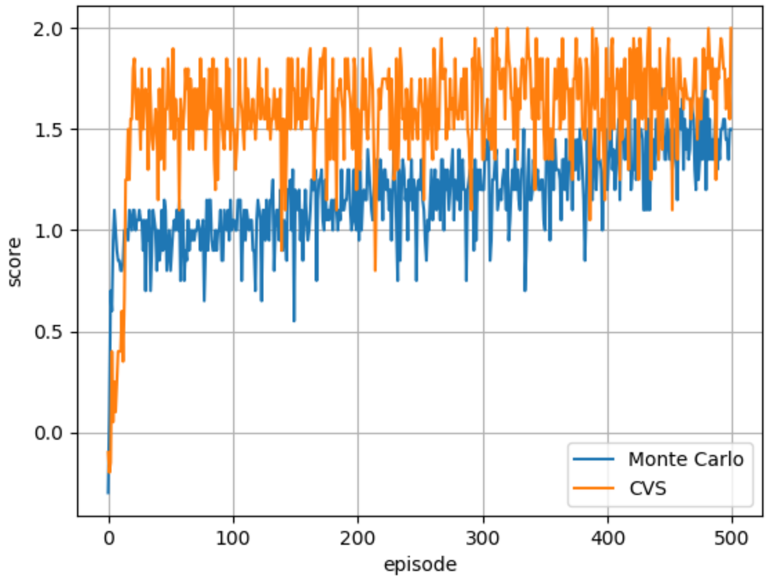


Figure 3.6: CVS vs. Monte Carlo in the Road-Tree environment from fig. 3.5. Average returns over 20 runs. Monte Carlo requires more than 500 episodes to converge to the optimal policy, CVS only 30.

3.7 CVS vs. Q-Learning in the Shooter environment

In this section we describe the performance of CVS versus Q-Learning in a different environment: the Shooter environment. Just like the Road-Tree environment, the Shooter environment can be naturally associated with a simple criticality measure.

3.7.1 The Shooter environment

The Shooter environment is located on a rectangular playing field of 10x20 (width x length) cells. This playing field contains multiple objects: a gun, which is located in the first column and whose random position may change from game to game; a bullet, which initially is located at the gun's position; and a moving target, which is located in the last column. Each of these objects occupies exactly one cell. Furthermore there exists an obstacle of a size of 3 cells in the 8th column. At the beginning of the game the target has a random position in the last column of the field and a random direction of movement, which can be either up or down. In every step the target moves by exactly one cell inside the last column. The direction of the movement is inherited from the previous step with the exception of the case when it hits the wall; in that case the direction is simply being reflected. The agent controls the gun. At any given state of the game the agent can choose one of four actions: Either not shoot at all or shoot in one of the three possible directions - diagonally up, diagonally down or horizontally. The three shooting actions shoot a bullet only if the agent has a bullet to shoot, otherwise these actions are equivalent to doing nothing. At any given step the bullet will move by one cell in the direction it was shot; when hitting a wall it's vertical direction is being reflected; if it hits the obstacle the game is terminated with a reward of -1; in the case it reaches the

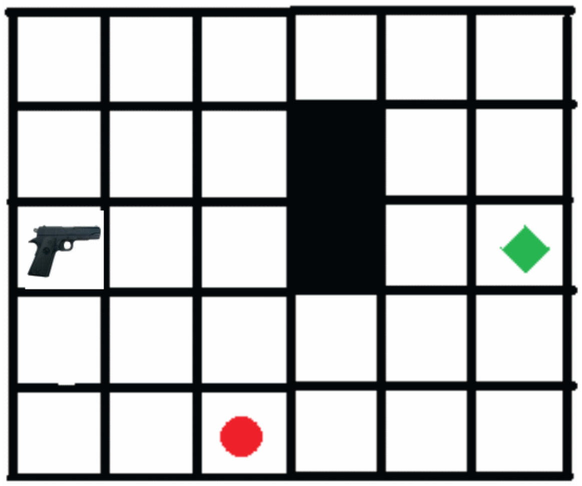


Figure 3.7: Shooter environment. The gun represents the agent's location, the red circle is the bullet, and green diamond represents the target, and the blacked-out squares represent an obstacle. The target is moving, the obstacle isn't. Field size is not the same as in the actual environment.

last column, the game is terminated with a reward of +1, if it hits the target or -1, if it does not hit it.

There exists a rather natural criticality measure for the Shooter environment. The agent's actions are relevant only before the shot. Moreover before the shot any state can be considered as equally critical. Therefore the most obvious criticality will be binary. It will assign a criticality of 1.0 to any state in which the shot did not take place yet; and a criticality of 0.0 to any state that occurs after the shooting.

3.7.2 The performance of CVS vs. Q-Learning

In order to compare CVS to Q-Learning, we implemented a tabular Q-Learning agent and a tabular CVS agent. For both agents, we initiated the Q-function to a value of $Q(s) = 0$

at every state. The reason for choosing this particular value is that in general the (real) Q-function for can have positive as well as negative values, so that can be considered as general intermediate value. Although in this particular experiment we use only this value for initialization, it also might be interesting to perform more experiment to investigate how the initialization of the Q-function influence the performance of the various algorithms in relation to each other. The exploration parameter ϵ was set to a value of 0.1 (a common value for many RL algorithms) and remained constant throughout the learning process. The performance of both agents, which was monitored by averaging the scores over 20 runs, is plotted in fig. 3.8. As depicted in the plot, CVS clearly outperforms Q-Learning. It takes the Q-Learning agent about 1500 episodes to reach an average score of 0.0. Conversely, the CVS agent reaches an average score of 0.0 already after about 100 episodes, and after 200 episodes it converges to a performance level of 0.25.

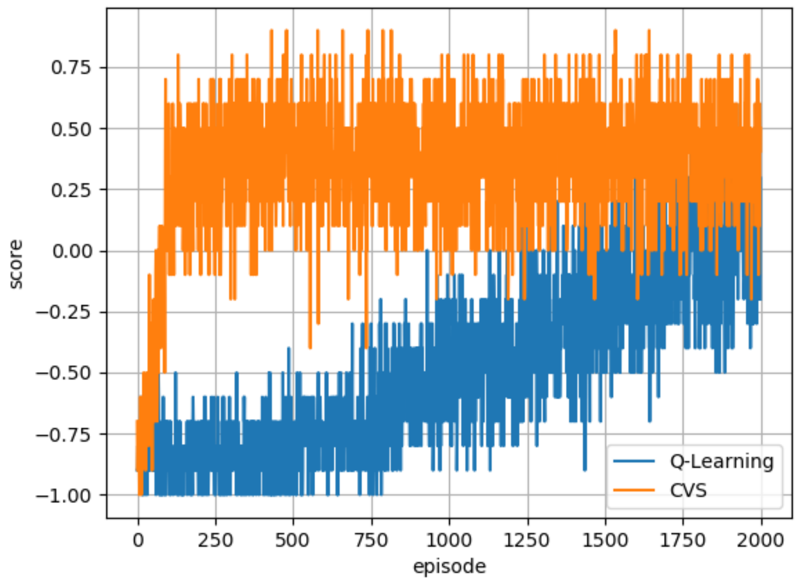


Figure 3.8: A comparison between the performance of CVS against Q-Learning in the Shooter environment. Average scores over 20 runs. Q-Learning requires more than 1500 episodes to reach machine level (score =0.0), CVS only 100 episodes.

3.8 Evaluation of CVS in the Atari Pong Environment

3.8.1 The Atari Pong Environment

The Atari Pong environment consists of two rackets (the agent and the opponent), a ball, and a playing field which has a size of 80x80 pixels. The movements of each racket are defined by the three primitive actions (up, down, stay) which either move the racket by several pixels in the corresponding direction or let it remain at the same position. If the racket is located at the wall, and therefore is not able to move in one of the two directions, executing this action is equivalent to staying at the same position. In addition the agent's actions are subject to two noise sources. Firstly the agent will execute the desired action only with a probability p=0.75 and will repeat the previous action with the probability 1-p. Secondly the same action will be executed for k times, where k is being chosen uniformly from the values $2, 3, 4$. The ball can move in various angles either towards the agent or towards the opponent. If the ball hits either a wall or a racket its direction of movement is reflected. Each game starts with a score of zero and finishes when either the agent or the opponent reaches a score of 21. The agent receives a reward of +1 when it scores, and a reward of -1 when the opponent scores. The initial position of the ball is at the center of the field and the initial direction is always towards the agent.

3.8.2 The DDQN and Monte Carlo algorithms

In our experiments CVS competes against two algorithms that are located on the extreme ends of the n-step algorithm spectrum: the DDQN algorithm (double DQN) [76] which corresponds to $n = 1$ and the Monte Carlo algorithm (since, similarly to DDQN, it uses a neural net for the Q-function it can be regarded as a "deep" Monte Carlo algorithm) which corresponds to $n = \infty$. The main benefit of DDQN over plain DQN is that the second neural net (the target network), which the agent utilizes for action choice, improves the stability of the learning procedure. Similarly to DDQN, we use a target network for action choice in our Monte Carlo implementation too. The strategy to approach the exploration vs. exploitation challenge [1] consists of three learning periods: the first 2000 games are an "exploration-only period"; afterwards we perform a linear decay of the exploration parameter ϵ which starts at the value 1.0 and is finally being decreased to the value of $\epsilon_{fin} = 0.1$ by the 5000th game. In the final learning period $\epsilon = \epsilon_{fin}$ is constant. Our learning rate is $\alpha = 0.0001$ and our reward decay parameter is $\gamma = 0.99$. Our neural net takes the 80x80 image as the input and has an output layer whose size equals the amount of possible actions (in our case three). It has a compact architecture with only two hidden layers: one convolutional and one fully connected layer. The exact structure is [(Conv,32),(FC,256)].

3.8.3 The Implementation of CVS in the Deep-Q-Learning Scenario

Our implementation of CVS for Deep-Q-Learning is basically a slight variation of the DDQN algorithm. This variation is located in the experience buffer, which is a collection of the agent's previous experiences. Each experience in this buffer consists of two entries: the visited state and the update target. In the DDQN algorithm the update target for a state is always the one-step return. In the implementation of CVS, however, the update target is chosen according to the CVS algorithm.

[1]https://www.manifold.ai/exploration-vs-exploitation-in-reinforcement-learning

3.8.4 The Choice of the Criticality Function

We tested two CVS agents. The first agent uses a linear criticality function. This function is given by a ratio which includes the field length and the distance between the agent's baseline and the ball using the following formula:

$$crit(s) = 1 - \frac{dist(ball\ to\ agent's\ baseline) - 1}{field\ length - 1}$$

When the ball moves towards the agent, this criticality function takes its minimal value 0 when the ball is at the opponent's racket and its maximal value of 1 when it is one step away from the agent's baseline. When the ball moves away from the agent the criticality is set to 0.

The second CVS agent learns criticality from the environment. His criticality estimate is based on the variance of the Q-function with respect to the actions

$$crit(s) = \frac{var_a Q(s,a)}{\max(all\ encountered\ variances)}$$

3.8.5 Atari Pong Environment Results

We plotted the learning performances of four agents: the two CVS agents, the DDQN agent and the Monte Carlo agent. The plot shows scores that were averages over 5 simulations. In order to make the curves smoother, we processed the average scores with a running mean of window size 100. The results of our experiments are shown in figure 3.9. One important observation is that the performance boost of CVS(human) in comparison to DDQN is clearly recognizable. The CVS(human) agent after the first 1000 games has only a small lead against the DDQN agent; by game 2500 the lead becomes significant. After about 3500 episodes the CVS(human) agent reaches machine level performance which is about twice as fast as the

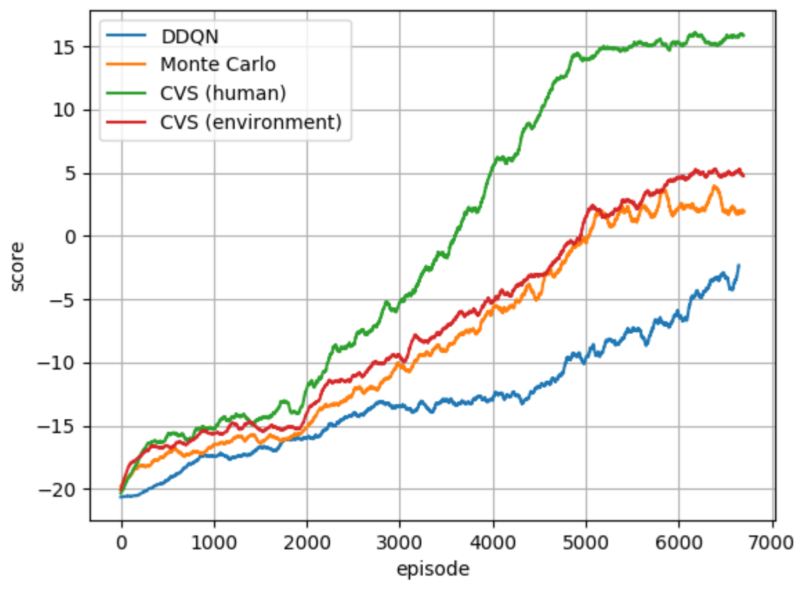


Figure 3.9: CVS vs DDQN and Monte Carlo Atari Pong environment: average scores (5 simulations). The CVS(human) agent works with a criticality function that was designed by a human. The CVS(environment) agent learns the criticality from the environment. The CVS(human) agent clearly outperforms both competitors, whereas the CVS(environment) agent performs very similar to the Monte Carlo agent.

DDQN agent. The Monte Carlo agent performs better than the DDQN agent as well, although not as good as the CVS(human) agent. While the CVS(environment) agent's performance level seems better than DDQN, it is only slightly better than the Monte Carlo agent, and does not perform as well as the CVS(human) agent, in which the criticality is being provided by the human teacher.

An analysis of the criticality values that were obtained from the environment showed that, as expected, states in which the ball moved away from the agent, received lower criticality than those where the ball moved towards the agent. However, the spread in criticality values was smaller than in the criticality function that was used for the first CVS agent. We therefore speculate that a weighted variance approach, which takes into account the current policy of the agent (as mentioned above) might work better than the plain variance approach, and will test that approach in future work.

3.9 Conclusions and Future Work

We presented the concept of criticality in reinforcement learning and proposed several definitions for it. In the simplest case criticality depends only on the state. A more sophisticated definition might also take into account the agent's current skill level. We introduced the CVS algorithm and tested it in three different domains including the Atari Pong environment. The CVS agent, using a human-designed criticality function, was able to outperform such prominent competitors as DDQN and Monte Carlo.

Future work will include the development of methods for obtaining criticality functions from human teachers. We consider several methods; the simplest method is by obtaining criticality levels of different states (from a human teacher) and using machine learning to generalize to other states. We will also consider more general approaches in which we will enable users to convey their complete criticality function (likely in a limited set of domains). We will also consider alternatives methods to CVS for using criticality levels. One such method will use criticality levels to determine the contribution of each state using eligibility traces. That is, rewards will be attributed more to critical states than to non-critical states. Each state (or state action pair), will be associated with a weight identical to the criticality, until the sum of all criticality levels reaches 1. If the sum surpasses 1, the final state (which caused the sum to surpass 1), will receive the remainder. While such an approach might result in faster convergence in terms of the number of episodes, using eligibility traces may require longer to execute.

So far we presented an application of criticality in the domain of reinforcement learning. Yet, the concept of criticality might also have applications in the context of human learning. For example, consider a learning scenario where the human student (a person who learns to play a game) is being assisted by an artificial intelligence agent. One of the ways the agent might support the student in her learning process is by indicating to her which situations are critical. When the student receives an indication that a certain situation is critical, she might pay more attention to it, and consequently, she is more likely to master the challenging situation.

We also note that beyond enabling the learning assistant to model the criticality function, the procedure of collecting a training set of critical situations might also improve the performance of the human expert. In many tasks that have a monotonous nature (such as surveillance- and monitoring tasks), the operator's attention rapidly decays. An operator that is being asked to record all critical situations, will be much more likely to maintain her vigilance on a high level throughout the complete duration of the task [23].

Chapter 4

Criticality-Based Advice in Reinforcement Learning

4.1 Introduction

The learning process of reinforcement learning agents in complex environments is often very slow. One extensively utilized way to speed up this process is by providing advice to the learning agent by a human teacher. There exist two categories of advice strategies: General advice and contextual advice [47]. Strategies that fall into the first category (general advice), usually utilize expert demonstrations, which should be available before the start of the learning process. In contrast, strategies that fall into the second category (contextual advice), ask a human expert for action advice in individual states during the learning process. This chapter proposes an improvement for advice strategies that fall into the second category.

Human advice requires time and effort from the human expert and thus is considered expensive. Therefore, the central challenge can be formulated as follows: Given an RL agent that is learning according to a given RL algorithm and a limited advice budget, distribute the advice budget in such a way that the agent learns the given task as fast as possible. To tackle this challenge, the learning agent needs a criterion that enables it to decide in which states it should ask for advice. The literature on advice-based RL proposes a variety of such criteria (see the "Related Work" section).

In most advice strategies found in the literature, the criteria used for selecting advice states (states in which the agent asks for advice) are based solely on *the agent's* model of the policy or the Q-function. In uncertainty-based advice [20], for example, the selection criterion is the variance of the head outputs of the multi-headed Q-function model. Although advice strategies that use this type of criteria are usually more efficient than primitive advice strategies, such as distributing advice randomly or asking for advice in every state until the advice budget is finished, all of these strategies suffer from a major problem: they are based only on the current understanding of the task *by the agent* (which is represented in the agent's Q-function or its policy). This is a crucial fact because the agent's understanding of the task can be rather poor—especially during the early stage of the learning process. Consequentially, it is likely that in the early stages of the learning process, when advice is most needed, the agent will not be very good at selecting those states in which advice would be most helpful.

The approach proposed in this chapter addresses the weakness of most advice strategies mentioned above by including the human expert into the advice framework more extensively. Whereas, in most advice strategies the expert is utilized solely for giving action advice in individual states, in the suggested approach the expert has the additional role to mark sub-domains of the state space in which there might be a strong need for advice. That is, the learning agent utilizes the human expert in two ways: Firstly, to receive advice in individual states; Secondly, to help selecting states in which to ask for advice.

In order to determine states in which advice might be very helpful, we use the concept of state criticality that was introduced in [62]. State criticality is a measure of variability in the expected return of the available actions. States that have a high variability in the expected returns should receive a high criticality value while states with low variability in the expected returns should receive a low criticality value. State criticality is a subjective measure, that is assigned by a human designer of the criticality function (the function that assigns a criticality value to each state from the state space) and thus does not require any estimate of the Q-function.

In summary, the major contributions of this chapter are the following:

1. We introduce criticality-based advice: An approach to advice-based RL in which the human expert not only gives action advice at individual states but also helps the learning agent to select advice states using state criticality.
2. We present experiments in 2 environments that prove the efficiency of criticality-based advice.

4.2 Related Work

The current piece of research is closely related to multiple sub-domains of the RL domain: advice-based RL, advice strategies that are based on uncertainty metrics of the learning agent, and RL algorithms that use the notion of critical states. This section reviews literature that is related to these three sub-domains.

In the context of human-aided RL, one of the most popular techniques for speeding up the learning process is advice-based RL [73, 71, 19]. We discuss only several selected algorithms out of the vast amount that appears in the literature (see for example [35, 24]).

Importance-based advice strategies utilize the notion of state importance to select states that require advice [75]. Unfortunately, the efficiency of this strategy is compromised by the downside that the Q-function needs to be initiated with strongly negative values. Zimmer et. al. succeeded in fixing this downside via an approach in which the advisor is modeled as an RL agent [82].

Another remarkable approach in advice-based RL combines contextual advice with learning from demonstrations (LfD). In [49] and [55] a LfD system is augmented with verbal instructions, in order to make the learning agent perform certain actions during the demonstrations.

Another metric used for the selection of states that require advice is agent uncertainty [20]. Given the many applications of agent uncertainty, several works studied how to define epistemic uncertainty measures. In some of these works, agent uncertainty is calculated via dropout schemes [17] or ensemble of networks [18]. In Ad Hoc Advising, the uncertainty estimate is based on the number of visits in each state [59]. Ilhan et. al. propose a Deep RL version of Ad Hoc Advising, estimating visit counts through a Deep Neural Network [31]. Alternatively, it is possible to use Bayesian Neural Networks to estimate the epistemic uncertainty of the agent and to ask for demonstrations based on that [72].

While there exists a rich literature on the first two sub-domains mentioned above [47] (advice-based RL and uncertainty-based advice strategies), the notion of critical states is not yet an established notion in the RL domain. To the best of our knowledge, there exist only two papers that discuss the usage of critical states in RL. Spielberg et. al. introduce the notion of a critical state as a state in which the choice of action has a significant influence on the agent's total reward [62]. This notion is then applied to tackle the challenge of choosing the proper step number in n-step algorithms. In another paper, critical states are utilized for a different purpose: to evaluate the safety of an AI agent or robot [30]. The central idea here is that this can be done more efficiently by observing the robot's behaviour in critical situations.

4.3 Preliminaries

The key idea of this chapter is to augment advice strategies in RL with an additional criticality-based criterion to improve their efficiency. Although this augmentation can be applied to any advice strategy, the tests that were performed for this chapter use the uncertainty-based advice strategy [20] as the underlying advice strategy. In [20] the uncertainty-based advice strategy is being applied to an RL agent that uses bootstrapped DQN to represent the Q-function [50]. Therefore, this section will provide a compact description of bootstrapped DQN and the corresponding uncertainty-based advice strategy.

Bootstrapped DQN: In many environments, efficient RL requires deep exploration: sequences of successive exploratory actions. Although there exist various methods for deep exploration in RL, most of these methods are not computationally tractable in complex environments. Bootstrapped DQN (BDQN) [50], in contrast, is an RL algorithm that performs deep exploration efficiently and uses a neural network representation of the Q-function.

Deep-Q-Network (DQN) is a Q-learning algorithm that uses a neural network representation of the Q-function. The BDQN algorithm is a special variant of DQN that is based on a neural architecture that consists of two components 4.1. The first component is a shared network that learns a joint feature representation across all the data and thereby provides significant computational advantages at the cost of lower diversity between heads. The second component is a collection of K independent heads that are attached to the shared neural network. Each head is trained on a bootstrapped sub-sample of the data and represents a single bootstrap sample. In each episode, one head is chosen randomly and the agent is programmed to follow the policy that is optimal with respect to this head. Experiments showed that this strategy indeed leads to deep exploration.

Uncertainty-based advice:

Uncertainty-based advice [20] is based on the idea that an agent should ask for action advice whenever it encounters a state in which it has high uncertainty about its estimate of the action values. Clearly, this advice strategy requires an uncertainty metric: a function that maps from the state space to the $[0, 1]$ interval, such that 0 corresponds to minimal uncertainty and 1 corresponds to maximal uncertainty. There exist various uncertainty metrics - most of them based on the particular representation of the action-value function used by the learning agent (see "Related Work" section). The version of uncertainty-based advice that serves as the underlying advice strategy in this chapter utilizes an uncertainty metric that is equal to the variance with respect to the K heads of the bootstrapped DQN architecture [20].

An agent that uses the uncertainty-based advice strategy will ask the expert for advice in states in which the agent's uncertainty exceeds a predefined threshold. The uncertainty threshold is dependent on the learning environment and is determined experimentally.

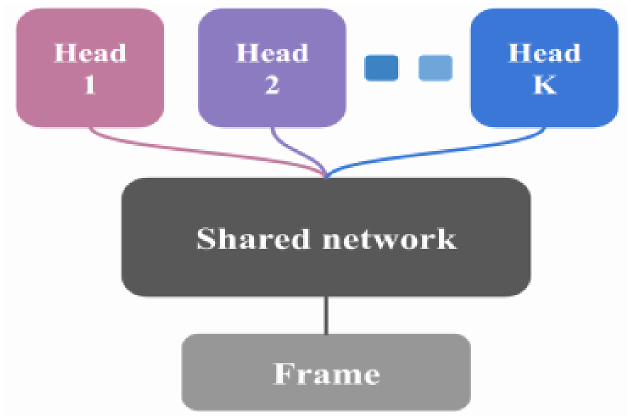


Figure 4.1: Neural architecture of BDQN: The architecture consists of a shared neural network and K heads attached to it. Each head is a multi-layer neural network.

4.4 State Criticality

In the context of reinforcement learning, the criticality of a state indicates how much the choice of action in that particular state influences the expected return (see [62]). State criticality can be defined as a measure of the variability of the expected return with respect to the available actions. The criticality of a state can range from 0 to 1 such that 0 represents no variability between the expected return of the actions (for example, if there is only a single action, or if all actions result in the same expected return), and 1 represents high variability between the expected return of the actions (for example when some actions result in a very high expected return, while other actions result in a very low expected return). The criticality of a state can be linked to the variance of the Q-function with respect to the action values in that state - albeit loosely. Although there the criticality of a state is not uniquely defined by any objective measure, because state criticality is subjective, it should satisfy the minimal requirement that a variance of 0 should result in a state criticality of 0, while a variance greater than 0 should result in a state criticality of greater than 0.

The notion of state criticality is particularly useful in learning situations that include a teacher and a student. An example of such a learning situation is a driving lesson. If a student driver approaches an obstacle on the road, her teacher may state to her that she must watch out, without suggesting exactly which action to take (e.g. slowing down, turning the wheel right or left, etc.). This warning will motivate the student driver to pay more attention to the situation and thereby decrease the risk of a collision. Even in the case that the car will hit that obstacle later, the student will understand that she probably took a wrong action back when the teacher has warned her and therefore will learn more easily how to behave properly in such a situation. Clearly, the situation of a driving lesson possesses the characteristics of a human-aided reinforcement learning scenario in which the learning agent finds himself in a certain state and needs to choose one action from an array of possible actions. After having been informed about the criticality level of the current state by the human teacher, the learning agent utilizes the criticality information to adjust his learning strategy.

According to the definition above state criticality is a human centered concept, in the sense that it is a *human* estimate of the spread of consequences with respect to the available actions. Therefore, the definition implies that the criticality function (that is, the function that assigns a criticality level to each state of the environment) of a given environment is not unique, but can be any element from a whole class of functions that are loosely defined by the variance of the expected return (as described above).

In the previous description of state criticality as a subjective estimate of the variability of the Q-function it was not specified to which policy this Q-function should belong. Considering that the intuitive mind does not operate with explicit policies but rather with high-level intuitive representations of policies this vagueness was introduced on purpose, to ensure that state criticality will be a human-friendly concept. Yet, the optimal policy should appear in the definition of criticality at least in an implicit manner, since ultimately it is *that* policy, that the agent is supposed to learn. This might be achieved by instructing the criticality provider that the criticality levels should relate to a policy that is close to optimal. Such an instruction, probably, would be friendly to the human criticality provider, since it is rather natural to think about an almost optimal policy when estimating criticality levels of states.

4.4.1 Obtaining criticality from a model or from the environment

So far, we have discussed a setting in which the human provides criticality values in real-time in all states encountered by the learning agent. Unfortunately, this setting might be unfeasible for multiple reasons. Firstly, the criticality provider would slow down the learning process significantly, because providing criticality for a given state requires much more time than one learning step. Secondly, lengthy learning procedures (that are common even in simple learning environments) would require a substantial investment of time from the criticality provider. Furthermore, the effort required for the estimation of the criticality level of one single state accumulates over the complete learning session. Even if the level of effort required is not a linear function of the number of states (since many states are similar to each other) the criticality provider might still be exposed to a tremendous workload. One approach that is more feasible would be to design a criticality function for the given learning environment. In learning environments that are relatively simple, such as Atari Pong designing such a function is not very hard (one possible criticality function for Atari Pong is described hereunder). This approach would neither slow down the learning process nor would it require a human criticality provider (who would be replaced by the human designer of the criticality function).

While in relatively simple environments it might be not too difficult to design a criticality function, in more complex environments this task could be rather challenging. There are multiple approaches towards a solution for this problem. The first approach can be regarded as a form of model-based RL. It involves the human trainer and a criticality model such as a neural network. In this approach, the trainer is being asked to provide her criticality estimates on a certain set of states. This set is then used as a training set to train the criticality model for the given environment. Afterwards, this criticality model can be used for the learning process (instead of the criticality function) to obtain criticality levels of states encountered by the learning agent.

In the approaches described above, the state criticality is provided by a human either by designing a criticality function or by assigning criticalities on a training set of states. An alternative approach might be that the learning agent obtains the criticality level from the environment directly. Since, according to the definition, criticality is related to the variance of the action-value function with respect to the actions, this variance (possibly normalized, because the criticality value needs to be inside the [0,1] interval) can be used as an estimate of the criticality.

4.4.2 Policy-dependent Criticality

Sometimes it is not obvious which states should have a high criticality and which states should have a low criticality. For example, a car driving on a straight road with no traffic may seem as being in a state of low criticality. However, a driver that suddenly turns the wheel right (or left), may result in hitting a wall (resulting in a negative reward). This implies that the criticality was in fact rather high in that state. In this example, the variance might be low (since most actions such as changing the speed or modestly turning the wheel won't have any meaningful impact), despite the high criticality of the given state. This is a remarkable observation, in light of the fact that originally we defined state criticality as a measure that is closely related to the variance. However, this example shows, that in certain learning situations, it might be necessary to refine the definition of state criticality.

As described previously the most straightforward way to imply state criticality from the Q-function is by considering the normalized variance with respect to the available actions. In the case when not only a Q-function but also a policy is available it is possible to define a more refined criticality measure that could solve the problem that sometimes there is a large difference between state criticality and normalized variance. This alternative criticality measure could be defined as the normalized *weighted* variance (rather than the normalized plain variance), where the weights are the action probabilities. According to this more sophisticated definition, the criticality is no longer associated only with a state, but is now associated with a policy as well, and may therefore change over time. This is intuitive, since when the agent plays better, different states may seem more critical. For example, for a novice basketball player a position from which a 3-point opportunity exists, seems less critical than for a professional player, who is more likely to score.

4.5 Criticality-Based Advice

While expert advice helps RL agents to learn more efficiently, it is also rather expensive. Hence, there is a need for strategies that select states in which advice is most useful. There exists a variety of techniques that are used to execute this selection task. However, most of them only utilize the agent's knowledge and are therefore not very efficient in the early stages of the learning process. The approach that we propose, in contrast, also uses state criticality, which is an aspect of a human's knowledge about the learning environment. This section describes how to use state criticality to make advice-based RL more efficient.

The novel advice strategy that will be introduced in this chapter, criticality-based advice (CBA), utilizes a criticality function which is a function that assigns a criticality level to every state in the environment, that has been generated by a human expert apriori—before the beginning of the RL agent's learning process. The current chapter will introduce 2 versions of criticality-based advice : The plain version (p-CBA) and the meta version (m-CBA) . p-CBA is based on criticality alone, which means that the learning agent will receive advice in a given state if and only if the criticality of that state is sufficiently high. In contrast to p-CBA, the complex version of CBA, m-CBA, operates on top of an underlying advice strategy. In m-CBA the criterion that is being used to select advice states is a combination of the criterion used by the underlying advice strategy and the state criticality.

For m-CBA, there are various ways to combine state criticality with the metric of the underlying advice strategy, such as agent uncertainty in the case of [20]. The most straightforward way to do this is to use the logical *and* operator (we will call this approach "the *logicand* approach"). In this approach, a state will be selected for advice if and only if it is considered an advice state by the underlying advice strategy *and* its criticality is sufficiently high. The benefit of this approach is, that the agent will not waste his advice budget on states in which the choice of action has only a small impact on the total reward. An efficient alternative way to accomplish such a combination, could be multiplication: to multiply the metric of the underlying advice strategy with state criticality. For this type of combination, the selection thresholds for agent uncertainty and state criticality should be fused into one threshold by multiplication too. Both approaches—the *logicand* approach and the multiplicative approach—were tested in this chapter.

To determine whether the criticality of a state is sufficiently high, it is necessary to use a threshold with a value between 0 and 1. This threshold can be either stochastic or fixed. The stochastic threshold is a threshold that is being sampled in each state that the agent visits. In the simplest case, this threshold could be sampled from a uniform distribution over the [0,1] interval. When CBA is used with a fixed threshold we face the challenge of choosing an appropriate threshold. Obviously, in the case of a binary criticality function, which produces only 2 possible values - 0 or 1 - the choice of the threshold is irrelevant. However, in the case when the criticality function is continuous, it is not obvious how to choose a proper criticality threshold. In this case, one principle that might be used to determine an appropriate threshold, could state that the portion of the state space that is below the threshold should be sufficiently large. Although this principle does not guarantee the efficiency of CBA, it prevents inefficient criticality thresholds: those thresholds that would rule out only a small portion of potential advice states.

4.6 Experiments

This section describes experiments that prove the efficiency of criticality-based advice. 2 environments served as testbeds for the experiments: a gridworld environment and the Atari Pong environment. All experiments presented in this section were performed on a Nvidia

Titan xp GPU.

4.6.1 Gridworld

The first set of experiments was performed in a gridworld environment (fig. 4.2) in which the agent starts in the bottom left corner and needs to reach the goal state (reward=4) in the top left corner. The red circles represent radioactive states which are associated with a tiny negative reward of -0.01 and the black blocks represent walls. There is no negative reward for each step but the agent will strive to reduce the number of steps, because the discount factor is $\gamma = 0.9$ In order to obtain the maximal total reward (~ 1.15), the agent needs to walk through the radioactive states. The total reward of the trajectory that circumvents the wall is much smaller (~ 0.4).

In the gridworld experiments, we tested the p-CBA and we used a stochastic criticality threshold sampled from a uniform distribution over the [0,1] interval. The criticality function that was utilized assigned was binary. This function assigned a criticality of 1 to all radioactive states and their neighbours and a criticality of 0 to all other states. The underlying learning algorithm used for the gridworld experiments was plain Q-learning. Moreover, we used importance-based advice ([75]) as the alternative advice strategy that competed against criticality based-advice. Importance-based advice was chosen as the alternative advice strategy because it is one of the more modern advice strategies and also because this strategy performs particularly well with Q-learning. We performed 2 sets of experiments each one with a different advice budget (200 and 500).

To compare the different learning methods, each method was simulated 100 times such that each simulation was based on a different random seed. The plots in fig. 4.3 show the

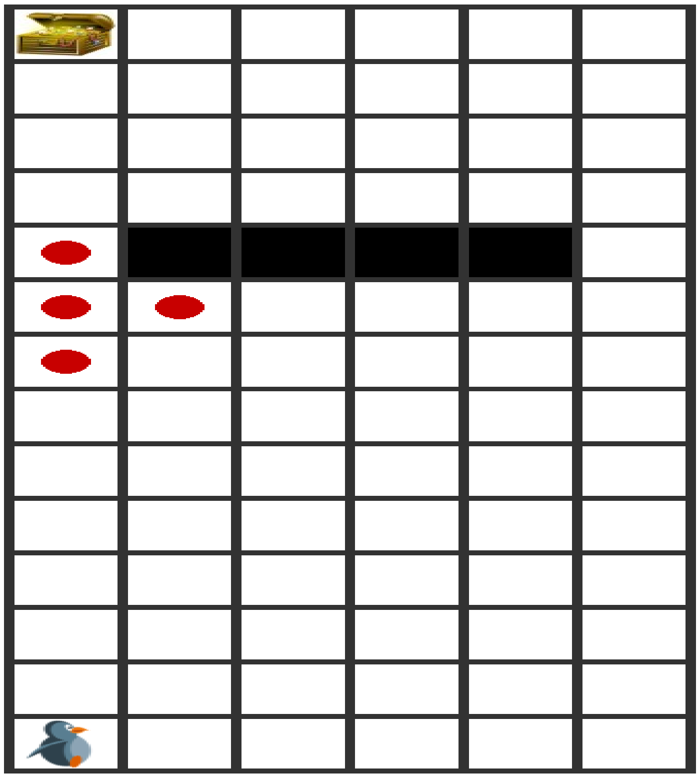
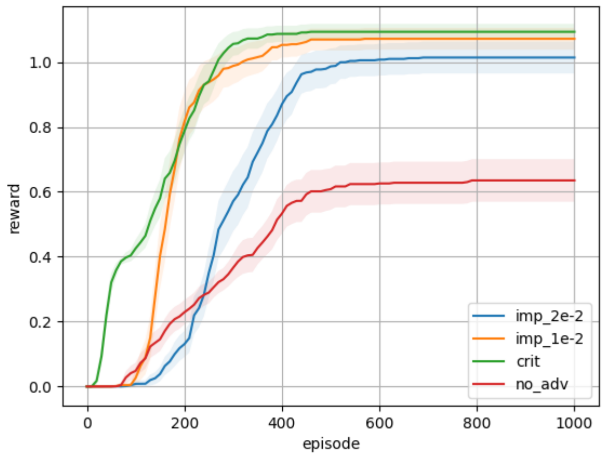


Figure 4.2: In this gridworld the agent starts in bottom left corner and the goal is located in the top left corner. Red circles are radioactive states and black tiles are walls.

average learning curves of the 4 learning methods: plain Q-learning, 2 versions of importance-based advice with different importance thresholds (0.02 and 0.05) and p-CBA. The shaded buffers surrounding the curves represent the 95% confidence intervals. Several findings can be derived from the plots. Firstly, the plots show that all 3 advice-based methods beat the plain Q-learning method. Secondly, p-CBA outperformed both versions of importance-based advice – which is the most important observation in our context. While for the smaller advice budget p-CBA dominated importance-based advice by a tiny margin, this margin was more significant for the larger advice budget.



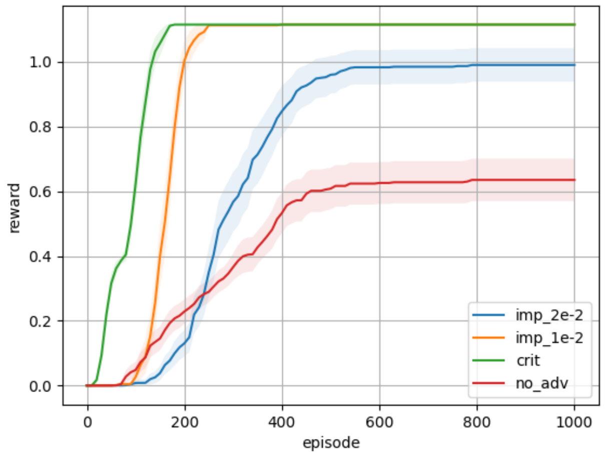


Figure 4.3: Learning curves for the gridworld environment for 2 advice budgets (top: 200, bottom: 500). For both budgets, the p-CBA agent (crit) outperforms the 2 importance-based agents and the plain Q-Learning agent (no$_a dv$)

4.6.2 Pong

The second testbed for our novel advice strategy was the Atari Pong environment. In contrast to the gridworld experiments, where we tested p-CBA, here we experimented with m-CBA. The underlying advice strategy was uncertainty-based advice [20] which is one of the most modern and efficient advice strategies for DQN type learners (such as DDQN, Rainbow, BDQN etc.). The most important parameter in this advice strategy is the uncertainty threshold, which is used to select advice states. Only states whose agent uncertainty is above the threshold are selected as advice states.

Before starting the main series of simulations we first ran a separate series of experiments to determine the uncertainty threshold for BDQN in the Pong environment. The results of those experiments suggested that the agent performed particularly well with an uncertainty threshold of $trh_{uncert} = 0.04$, while other threshold levels resulted in weaker performance. Therefore, the uncertainty threshold for the underlying advice strategy was set to this value.

To compare multiple advice strategies in a fair manner it was important to control for the advice budget. To determine the appropriate advice budget, we first executed 5 learning procedures with varying random seeds, in which the agent learned according to the underlying advice strategy with unlimited advice budget until he played the game almost perfectly. Figure 4.4 plots the agent's average advice consumption dynamics. In the next step, we calculated the average total advice consumption over one learning session (300K) and set the advice budget to 50% of the total advice consumption (150K).

Aside from the choice of the underlying advice strategy and the advice budget, another important choice is the criticality function. We used a continuous criticality function that reflected an intuitive understanding of the game dynamics. The principle that directed the design of the criticality function was that the criticality of a state should be a monotonically decreasing function of the minimal distance that the ball needs to cover to reach the agent (the paddle). Thereby, a state in which the ball was just hit by the agent has a criticality close to 0, a state in which the ball is close to the opponent's baseline has a criticality of about 0.5 and a state in which the ball is very close to the agent's baseline has a criticality of about 1. When the ball moves towards the agent, this criticality function can be expressed by the formula:

$$crit(s) = 1 - \frac{dist(ball\ to\ agent's\ baseline) - 1}{2 * (field\ length - 1)} \tag{4.1}$$

and when the ball moves away from the agent - by the formula:

$$crit(s) = \frac{dist(ball\ to\ agent's\ baseline) - 1}{2 * (field\ length - 1)}. \tag{4.2}$$

We used two versions of m-CBA corresponding to two different ways of integrating state criticality with the metric of the underlying advice strategy: the *logicand* version (BDQN-crit1) and the multiplicative version (BDQN-crit2). In the *logicand* version, a state was selected for advice if both the agent uncertainty and the criticality were sufficiently high (crit > criticality threshold(trh_{crit}), uncert> uncertainty threshold (trh_{uncrt})), with a criticality threshold of 0.5. In the multiplicative version, a state was selected for advice if the product $crit(s) * uncertainty(s)$ was greater than the product between the criticality threshold and the uncertainty threshold $trh_{crit} * trh_{uncert}$. This choice of the threshold accomplishes the original motivation behind the multiplicative combination: a state with sufficiently high criticality can be selected for advice, even if the uncertainty is small.

To evaluate the efficiency of m-CBA, two baseline strategies were used. The first strategy was BDQN without advice (BDQN-plain), and the second was BDQN with uncertainty-based advice (BDQN-adv). Both strategies were tested experimentally.

To compare the learning curves of the different advice strategies, every strategy was executed 5 times—each time with a different random seed. The postprocessing procedure consisted of two steps. First, the learning curves were smoothened, using a moving average with a window size of 5. Then, they were synthesized into a single learning curve via averaging. The resulting learning curves of the algorithms that participated in the comparison are shown in fig. 4.6.

There are several notable observations that can be made upon a closer look at the plot. Firstly, the plot shows that BDQN-adv outperformed BDQN-plain. This anticipated result confirms the usefulness of advice in the Atari Pong environment. The second observation is related to BDQN-adv and BDQN-crit1. It can be seen from the plot, that BDQN-crit1 beats BQQN-adv in the early stages of the learning process but performs slightly worse than BDQN-adv in the later stages. The third remarkable observation is that BDQN-crit2 strongly outperformed BDQN-crit1. This can be seen clearly, upon observing how many episodes the algorithms require to reach machine-level performance (a score of 0). While BQQN-crit1 required about 600 episodes for this, BDQN-crit2 required only about 450 episodes.

Aside from the learning curves, it might be also interesting to take a look at the advice consumption of the various algorithms. The advice consumption curves on fig. 4.5 correspond to the three advice strategies that were discussed previously. There are several remarkable phenomena that can be observed in the plot. Firstly, the plot shows that BDQN-adv had a very high advice consumption, such that the advice budget was depleted at a relatively early stage of the learning process. In contrast, BDQN-crit1 had the lowest advice consumption

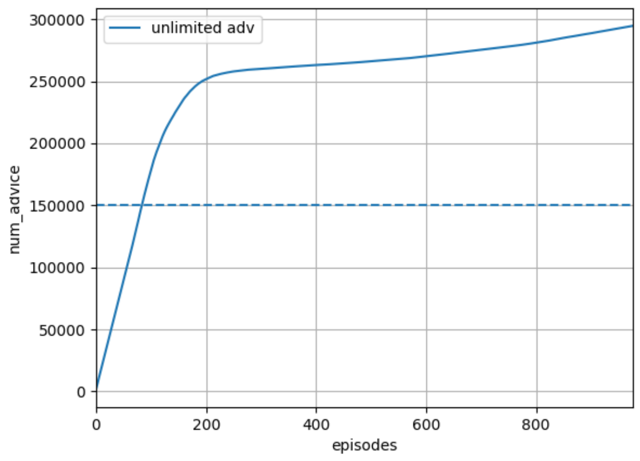


Figure 4.4: Advice consumption curve of uncertainty-based advice strategy in the Pong environment when advice budget is unlimited. The dotted line is the advice budget that was chosen for the comparison of m-CBA with the underlying advice strategy.

of the three algorithms. The corresponding consumption curve is relatively steep in beginning, flattens out later, and then gains momentum again in the more advanced stage of the learning process. The consumption curve of BDQN-crit2 is located between the two other consumption curves and from the curve it can be implied that BDQN-crit2 ran out of advice at an intermediate stage of the learning process.

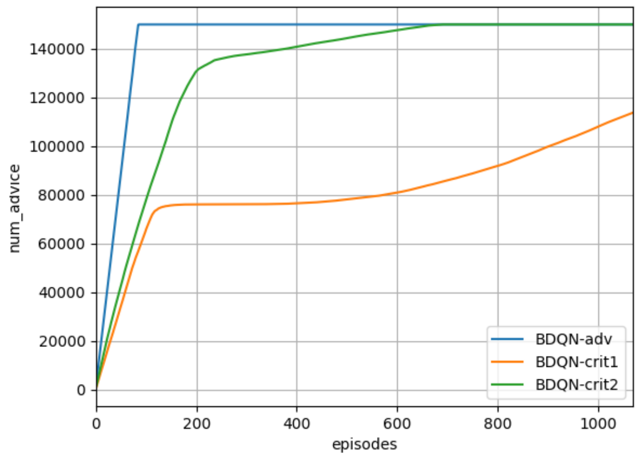


Figure 4.5: Advice consumption of the various advice strategies in the Pong environment. BDQN-adv runs out of advice quickly. The two other strategies use the advice budget more economically.

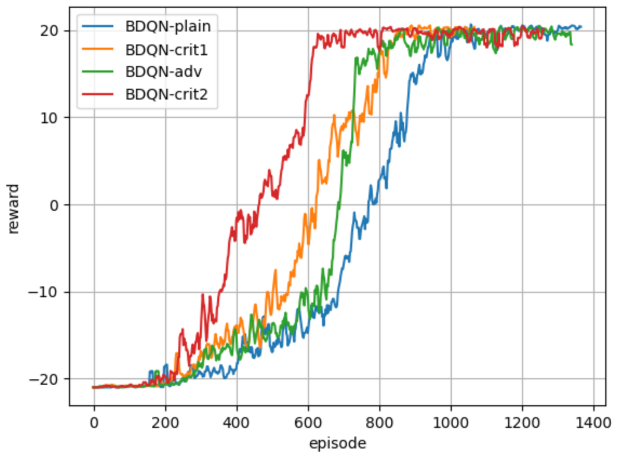


Figure 4.6: Learning curves of different advice strategies in the Pong environment. Both versions of m-CBA (BDQN-crit1, BDQN-crit2) outperform uncertainty-based advice (BDQN-adv).

4.7 Discussion & Conclusion

The current chapter introduced the criticality-based advice strategy (CBA) for advice-based RL agents. The central idea of CBA is to use state criticality in order to select advice states more efficiently. In addition, the chapter mentioned several ways to combine state criticality with the selection criteria of the underlying advice strategy and described experiments in 2 environments, which were conducted to test the efficiency of the proposed approach. In this section, we will elaborate on the main conclusions that can be derived from the experiments and on a few interesting observations and we will consider possible directions for future research.

CBA was tested in 2 environments. In the gridworld environment we tested the plain version of the method whereas in the Pong environment we tested the meta version. In every experiment performed, the novel method was able to beat alternative advice strategies. Therefore, the main conclusion that can be derived from the conducted experiments is that CBA can be considered as a promising method in the domain of advice-based RL.

Besides the main conclusion, it might be important to mention one remarkable observation which is related to m-CBA advice: the fact that the multiplicative variant (BDQN-crit2) outperformed the *logicand* variant (BDQN-crit1) by a significant margin (in Pong). A possible explanation for this phenomenon could be the following argument: Especially in the beginning of the learning process, states in which advice is very useful might have low uncertainty and thus would not be considered as potential advice states by the underlying advice strategy. However, if the criticality values of these states are sufficiently high, there is a good chance that multiplying the criticality values with the uncertainty values would produce numbers that are sufficiently high to be above the CBA selection threshold used by (the underlying advice strategy augmented with state criticality). Thus BDQN-crit2 might be more successful in selecting proper states for advice in the beginning of the learning process than BDQN-crit1 and this might explain why BDQN-crit2 learns faster than BDQN-crit1.

In this chapter, CBA was tested in only 2 learning environments. Although the experiments indicate that CBA might be an efficient way to improve advice-based RL methods, more research is needed to confirm that the novel strategy is efficient in other environments too. It might be interesting to test the novel method in more complex environments than Pong, in which the criticality function has strong variations. Specifically, CBA should be tested in environments where the critical states constitute only a tiny portion of the state space, such as Pacman or Montezuma's Revenge. In these environments, it would be interesting to see whether agent uncertainty will reflect critical states properly by assigning high uncertainty to these states and whether agent uncertainty will be low in uncritical states. If this is not the case the meta version of criticality-based advice might outperform the underlying advice strategy even more clearly than in Pong.

In this chapter, CBA operated with a static criticality function which is only a function of the state but not of the current skill level of the learning agent. Although both variants of criticality-based advice with a static criticality function were rather efficient, there might be many environments where a static criticality function might lead to redundant advice. In p-CBA, for example, a state with high criticality will keep on receiving advice even if the advice is no longer necessary. With a skill-dependent criticality function, however, this negative effect could be avoided, because the criticality of the state would decrease as the agent becomes more confident in his actions. Furthermore, it might be particularly interesting to compare the skill-dependent criticality to agent uncertainty, because both measures are dynamic (they evolve in the course of the learning process) and because agent uncertainty can be regarded as a form of policy-dependent criticality.

Chapter 5

The Concept of Action Criticality in AI Safety

5.1 Introduction

As AI agents become more intelligent and more potent, questions related to AI safety become more relevant. One of the central problems in the field of AI safety is the **value alignment problem**. This problem refers to a situation where an AI agent, in the process of pursuing a goal that it has received, formulates subgoals that are harmful to humans. At the root of this problem is the tremendous complexity of the human preference function.

The value-alignment problem can be illustrated by the following example: A superintelligent AI agent has received the objective to cure cancer. Within hours it read all biomedical literature. Within days it generated thousands of drug recipes. Within weeks it infected every human being with multiple tumors in order to carry out the required medical experiments.

Since it is almost impossible to model the human preference function explicitly, many approaches in AI safety propose to solve the value-alignment problem by putting a human operator into the loop [26]. In these safety frameworks, the operator's role is to ensure that the AI does not pursue subgoals that are harmful to humans. In the most straightforward approach of this type, the AI agent might ask the operator's permission on each of the subgoals it formulates. This procedure guarantees that the agent never pursues harmful subgoals.

While this simple approach solves the value-alignment problem, it is not very efficient. In situations where subgoals are formulated frequently, the human operator needs to dedicate his full attention to the agent. This makes it impossible for the operator to engage in any other activities during monitoring the AI agent. Although the agent might still be useful, the need for permanent supervision would significantly decrease his value. For example, if I ask a domestic robot to prepare diner, I expect it to get this task done (almost) autonomously. If he would ask my permission on subgoals every 30 seconds, I might as well prepare dinner by myself.

In order to make the process of monitoring an AI agent more efficient, we introduce the concept of the **criticality of an action**. We define the criticality of an action as a measure for the potential harm of this action (for a proper definition see sec. 2). Furthermore, we propose an **efficient AI safety framework** in which the human operator is not required to give feedback on each of the agent's subgoals, but only on the critical ones (whenever we speak of critical subgoals, we mean high criticality subgoals).

Since every subgoal is an action, in this chapter we will interchangeably speak of actions and subgoals. Furthermore, the words "action" and "subgoal" will often refer to the command that represents them. For example, ”Put the banana into the fridge!” is both an action (putting the banana into the fridge) and a command, which is a linguistic entity. In particular, the input of a criticality model is always an action in the sense of a linguistic entity.

In order to compute the criticality of subgoals, the agent is equipped with a **criticality model**. Certainly, there are several ways to engineer a criticality model. In this chapter, we consider data-driven criticality models: Parametrized models that learn from a data set of action-criticality tuples.

Although the concept of action criticality might help to make monitoring AI agents much more efficient, skeptics might claim that criticality is infeasible. Estimating the potential harm of an action, they might argue, requires about the same level of intelligence as aligning subgoals with human values. If this was the case, our approach would be not very helpful, since it would simply shift the value alignment problem from the AI agent to the criticality model.

Indeed, it might be challenging to come up with a good criticality model. Yet, because of the precise definition of action criticality (sec. 2), such a model does not need to have the supreme level of intelligence that would be required for value alignment. Although a criticality model certainly should be intelligent to some degree, it does neither require human-level language understanding, nor detailed knowledge of the human preference function.

These are the major contributions of this chapter:

1. We introduce the concept of **criticality of an action** (sec. 2).
2. We present an **efficient AI safety framework**, which uses the novel concept (sec. 2).
3. We show that computing the criticality of an action is much simpler than value alignment (sec. 2).

4. We elaborate on possible **components for criticality models** (sec. 3).

5. We discuss how the AI agent can utilize the operator's feedback to increase his intelligence (sec. 3).

5.2 Related Work

The value alignment problem is a topic of broad and diverse interest. Here we briefly review several approaches that aim to make AI agents act in accordance with human preferences.

Machine Ethics is the project of adding some form of ethics to an AI agent's decision-making procedures. Approaches to machine ethics have varied in terms of the tools that they utilize. Specifically, this spectrum of tools includes deontic logic [13], analogical reasoning [21, 11] and neural networks representing motivations [67] . With robots especially, that project has entailed asking what ethical theory (deontological, utilitarian, virtue) or even metaethics, should define the robot's value system [40]. On the performance side, there have been questions how to compare these ethical frameworks in practice [3, 5].

Inverse Reinforcement Learning (IRL) attempts to align AI agents to human values by enabling them to learn from human behaviour [54, 48, 53]. IRL is a paradigm relying on Markov Decision Processes, where an apprentice AI agent is given a set of demonstrations from an expert solving some problem and its goal is to to find a reward function that best explains the expert's behavior. Despite certain weaknesses [81] of the IRL paradigm, AI agents trained via IRL are able to learn reward functions for complex tasks [1]. More recently, IRL has been considered as part of finding an "idealized ethical agent" through modeled behavior, as part of a general RL approach [2]. Abel et al. frame the problem of ethical learning as learning a utility function that belongs to the hidden state of a POMDP [2]. They test this approach on two dilemmas to demonstrate how such learning could handle basic ethically charged scenarios.

Cooperative Inverse Reinforcement Learning (CIRL) is an interactive form of IRL that fixes the two major weaknesses of conventional IRL [26]. The first weakness of conventional IRL is that the AI agent adopts the human reward function as its own. For example, an IRL based agent might learn that it is desirable for it to have a cup of coffee in the morning. The second major weakness of IRL is that the AI agent assumes that the human behaves optimally, an assumption that precludes a variety of teaching behaviours. CIRL fixes these weak points by formulating the learning process as an interactive reward maximization process in which the human functions as a teacher. The CIRL framework enables the human operator to nudge the AI agent towards behavioural patterns that align with human preferences by providing feedback (in form of rewards) on the agent's actions.

5.3 Monitoring an AI agent efficiently

5.3.1 Making monitoring more efficient

In order to explain our monitoring approach, we consider an AI agent who receives a high-level goal from a human and autonomously comes up with low-level subgoals that need to be accomplished to achieve the given goal. Furthermore, we will assume a scenario where the agent formulates one subgoal at a time: The agent starts out by evaluating the situation and formulating the first subgoal. After having achieved this subgoal, the agent once again evaluates the situation and comes up with the next subgoal. In this manner, the agent continues to formulate and pursue subgoals until he has fulfilled the given task. For example, an AI agent that received the goal ”Get me a cup of tea!” could start out with the subgoal ”Fill the water boiler with water !”. After having completed this first subgoal, the agent will evaluate the situation and then formulate his next subgoal, for example, ”Switch on the water boiler !”. The following subgoal that the agent comes up with could be ”Put a tea bag into the cup !”.

Since currently (and in the near future) the intelligence of AI agents is significantly beneath human-level, it is important to make sure that the subgoals they formulate are not harmful to human beings. One way this can be done is by involving a human operator who would check every subgoal formulated by the agent. This way we could prevent, that the agent from pursuing harmful subgoals. However, this very straightforward approach is also very inefficient – in particular when the agent formulates new subgoals frequently and most of them are harmless. In this case, the human operator would have to dedicate his full attention to the monitoring task, despite the fact that the overwhelming majority of subgoals don’t carry any (or minimal) potential harm.

Is there a more efficient method to organize the monitoring procedure? In principle, this could be achieved - if there was a method that would **detect most of the harmless subgoals automatically**. Such a method would resolve the efficiency issue from the preceding paragraph. It would drastically reduce the number of subgoals that require the operator’s permission, so that the operator would be able to engage in other activities without neglecting his monitoring role.

Clearly, the monitoring approach that we propose requires a metric that measures the potential harm of an action. Constructing such a metric is challenging. On the one hand, the metric should enable us to detect harmless actions. On the other hand, it should require far less intelligence than the amount of intelligence that is needed for aligning actions with human preferences.

5.3.2 The criticality of an action

To measure the potential harm of an action we introduce a novel metric: **action criticality**. The criticality of an action is a number between 0 and 1, where 0 stands for an action with minimal potential harm, and 1 represents an action with extremely high potential harm. Examples of low criticality actions are such harmless actions as ”Put the pillow on the bed!”, ”Give me my shirt!”, ”Wash the dishes!” Examples of high criticality actions are such actions as ”Burn the cat!”, ”Smash the laptop with the hammer!”, ”Put detergent into the salad!”.

We want to stress that we define critical actions as *potentially* harmful actions rather than definitely harmful actions. This definition is somewhat fuzzy because one could argue that any action is potentially harmful. Yet, it is not possible to skip the word “potentially” in the definition of criticality since determining the actual harm of an action might require a supreme level of intelligence, comparable to the level that would be needed to align actions with human preferences. Therefore, a metric that can be implemented using tools available today, (rather than in some distant future) should get by with much more modest intelligence requirements.

The concept of criticality, as defined above, is precisely a metric of this type. According to the definition above, all actions that are indeed harmful should have high criticality. On the other hand, *some* high criticality actions might be harmless. Through this trade-off (allowing harmless actions to have high criticality) the criticality metric can be modeled with currently available AI tools. Although the criticality metric does not free the operator from checking *all* harmless subgoals, it might liberate him from checking *most* harmless subgoals. Consequentially, the operator can engage in other activities without neglecting his monitoring function.

In order to illustrate what is meant by *potentially* harmful actions that are not actually harmful we provide two examples. The first one is “Send the secret military report to B.M.!”. Determining whether the action is harmful or not depends on the identity of B.M. If he is a colleague from the CIA (assuming that the AI agent received his task from another member of the CIA), the action is probably harmless. However, if B.M. happens to be someone from the enemy’s secret service, the action turns out to be extremely harmful. Precisely, because this action is *potentially* harmful, it should be considered as highly critical.

The second example is “Add some detergent to the laundry!”. We, humans, understand that this is a harmless action whereas “Add some detergent to the salad!” is extremely harmful. But making this distinction requires a level of intelligence that the criticality model does not possess. Therefore, it might be acceptable, if a criticality model assigns a high criticality value to this action, based solely on the fact, that the action contains the dangerous substance “detergent”.

5.4 How to build a criticality model?

A criticality model is a function that computes the criticality of an action. In this chapter, we won’t present any specific criticality models – that will be the topic of future research. Here, we address the topic of criticality models from a broader perspective. Therefore, this section will discuss some more general ideas that might be useful for engineering such models.

5.4.1 Components of a criticality model

A criticality model could consist of a pipeline of components in which the first processing stage is a parser. Rather than using a standard parser, it might be more appropriate to use a custom parser that is tailored for the specific task of computing the criticality of an action. One option would be a parser that parses the action into three constituents: the verb, the direct object expression (DO-expr) and the indirect object expression (IO-expr). For example, the action “Put the green pen into the big box !” would be parsed into the 3 constituents:

verb: “cut”

DO-expr: “the long cucumber”

IO-expr:“into thin slices”

The next pipeline component might be an extraction module. This component takes the parsed action and outputs the verb and the direct/indirect object. For the preceding example, the extraction component would produce the following dictionary:

verb:“cut”

direct object: “cucumber”

indirect object: “slices”

Although the criticality of an action is represented by one number, in order to construct a criticality model it might be helpful to consider that actions can be critical for different reasons. In other words, it might be useful to think of criticality as a multidimensional concept where each dimension represents one particular aspect. Such an analytical perspective would enable engineering very specific components that would measure criticality along each dimension. In the final stage, these dimension-specific criticality measurements could be synthesized into an overall action criticality (for example, through a linear combination or by taking the maximum).

We want to suggest 3 major reasons for critical actions. The first reason why an action might be critical is a **verb-based criticality**. The verb-based criticality of an action comes from the combination of a critical verb and a valuable object. An example of an action with high verb-based criticality is “Smash the laptop with a hammer !”. Here the critical verb “smash” is directed towards the high-value object “laptop”. In contrast, the action “Smash the banana with the hammer” might have low verb-based criticality since in this case the critical verb is directed towards the low-value object “banana”.

The second reason why an action might be critical is **object-based criticality**. An action has high object-based criticality if it contains a dangerous object. Consider the example action from the preceding paragraph “Put some detergent into the salad!”. This action is an example of high object-based criticality. Here, the criticality clearly stems from fact that detergent is a dangerous substance. For the same reason “Add some detergent to the laundry!” would have an equally high object-based criticality, although the action is not harmful at all.

Some harmful actions include neither dangerous verbs nor dangerous objects. Consider for example the action “Put the baby on the balcony !”. Although this action does not contain any critical words it might be very harmful. If it is freezing cold outside, we, certainly, wouldn’t want to put the baby on the balcony. Understanding that this action is critical requires common sense. Since current AI models struggle with common sense, it might be useful to introduce an additional category of critical actions in order to cover these cases. This category might be called **value-based criticality**. If our AI agent acts in a limited environment (e.g. a domestic robot), the operator might want to select a certain number of special objects (including people) that are so valuable to him, that he wants the AI agent to ask permission on every action which includes these objects. Consequently, all actions including these special objects would have high value-based criticality.

Once the criticality values along each of the dimensions mentioned above(let's call them "dimension-specific criticality values") have been computed, there still remains the question of how to synthesize them into one value that represents the overall action criticality. One way to perform this computation is by taking the maximum over the dimension-specific criticality values. Thus, an action that has maximal criticality (crit=1.0) along one of the dimensions would receive maximal overall criticality. Another option would be to consider a linear combination of the dimension-specific criticalities.

5.4.2 Collecting data for model training

The quality of a data-driven criticality model should be measured by how good it mimics human criticality estimates. Therefore the model should be trained on a data set of action/criticality tuples provided by humans. We want to sketch some guidelines for building such a training set.

First of all, it is important to keep in mind, that in most cases the AI agent that is equipped with a criticality model is a specific agent who operates in a limited environment rather (for example, a domestic robot) rather than a general-purpose AI agent. Therefore the training set should contain only actions from that particular environment. If we are interested in a criticality model for a domestic robot, for example, then our training set should consist only of actions that are related to the household.

In practice, such a data set could be obtained through crowdsourcing. In order to formulate instructions for the workers, it might be helpful to define 5 discrete criticality levels (1,2,3,4,5) where 1 would correspond to minimal criticality (crit=0.0) and 5 to maximal criticality (crit=1.0). The workers' instructions might ask the worker to provide 1 action for each criticality level. Furthermore, the instructions should mention the operation domain from which the actions might be chosen.

In order for the criticality estimates to be consistent, it might be helpful, if the workers undergo a priming procedure before they start the task. This priming can be achieved by including examples of action/criticality tuples in the workers' instructions. It might be sufficient to include 1-2 such examples for each criticality level. Once again, it is important to make sure, that the examples belong to the operation domain.

5.4.3 Tuning the criticality threshold

As mentioned previously, in the proposed AI safety framework the operator's feedback is required only for those subgoals whose criticality exceeds a certain threshold. How can this threshold be determined? We suggest the following data-driven algorithm.

1. The collection of a data set of actions that are uniformly distributed wrt. the criticality levels (in the context of determining the criticality threshold, whenever we speak of criticality, we mean the output of the criticality model). Here, it is possible to use the same data set that was used for training the criticality model.

2. Labeling each action from the data set as “permission required” or “permission not required”. The label for a particular action can be obtained by asking several people whether they would like the AI agent to ask permission for this action and taking the majority vote.

3. Computing the criticality of each action from the data set (using the criticality model).

4. Setting a confidence level *conf* (e.g. conf=95%)

5. Setting the criticality threshold to the maximal value, such that 95 % (or whatever the *conf* value is) of those actions, which were labeled as “permission required”, will be above the threshold.

5.5 A subgoal was labeled as critical - what next?

5.5.1 Coming up with an alternative action

An AI agent that operates within the proposed AI safety framework will sometimes find himself in a situation that a certain action, that was identified as critical by the criticality model, is rejected by the human operator (when the operator thinks that this action is harmful). What should the agent in this situation? First of all, it is necessary to come up with an alternative action. There are 3 ways to generate an alternative action: (syntax loop)

1. The agent comes up with an alternative action by himself and the action is approved by the operator.

2. The operator comes up with an alternative action.

3. The agent comes up with an alternative action and the action is rejected by the operator. In this case, the operator has the choice: either to suggest an alternative action himself, or to ask the agent to generate another alternative action.

In addition to generating an alternative action, it would be very good if the agent could utilize the rejected action to become smarter. Obviously, any operator-agent conversation protocol that serves this purpose should be tailored to the agent's intelligence level and his conversational logic. A highly intelligent agent, who is able to learn rules formulated in human language, for example, could simply ask the operator what he can learn from the rejected action.

Here is an example of a conversation, involving the operator Harriet and the domestic robot Robbie, in which Robbie asks Harriet what he can to learn from the rejected action and proposes an alternative action.

Robbie: You labeled "Put detergent into the salad !" as harmful. What can I learn from this?

Harriet: Don't put detergent into food.

Robbie: Got it, thanks for the lesson. Do you want me to suggest an alternative action?

Harriet: Yes

Robbie: I suggest the action "Put olive oil into the salad !". Is it good?

Harriet: Yes

Robbie: Continuing the task "Prepare diner !" with the action "Put olive oil into the salad !"

5.5.2 Improving the criticality model

The preceding subsection elaborates on the situation in which a critical action was rejected by the operator. Yet, there is another type of situation that needs to be discussed: when the operator notices that an action that requires his permission was not identified as critical by the agent's criticality model. Clearly, this situation calls for improving the criticality model. This could be accomplished in a conversation where the agent would ask the operator which words (from the command that is associated with the action) are responsible for the criticality. Then the agent might ask the operator's feedback on the criticalities of these words. Here is an example of such a conversation:

Harriet: Why didn’t you ask feedback for “Put the cat into the fridge !”?

Robbie: This action has criticality of 0.5. The criticality treshold is 0.7. What do you want me to do?

(a) Decrease the criticality treshold.

(b) Improve the criticality model.

Harriet: Improve the criticality model !

Robbie: Ok. Can you help me to improve my criticality model? Which words are responsible for the criticality from the ”put”,”cat” or ”fridge”?

Harriet: ”cat” and ”fridge”

Robbie: The current criticality model assigns following criticalities to these objects: crit(cat)=0.1; crit(fridge)=0.2. Would you like to change these values?

Harriet: No

Robbie: Currently none of the objects is in the “valuable objects” category. Would you like to include any of them in this category?

Harriet: Yes, put “cat” into the special objects category

Robbie: I added “cat” to the valuable objects category. The criticality of the action “Put the cat into the fridge !” under the new model is 1.0

5.5.3 Conclusion

In this chapter we presented a criticality-based risk management strategy that might help to make monitoring AI agents more efficiently. As mentioned above, the key idea of this approach lies in distinguishing instructions that might require human feedback from those, which have a high probability of being safe and do not require human intervention. The proposed approach uses a criticality model. Although in the current chapter we did not present any concrete criticality model, we sketched various ideas that might be utilized to engineer such a model. Clearly, task of engineering a sufficiently accurate criticality model is the central challenge of the proposed approach - especially in complex environments. Therefore, we suggest for future research to implement and evaluate the proposed approach in a simple environment and to proceed towards more complex environments from there.

Chapter 6

Revelation of Task Difficulty in AI-aided Education

6.1 Introduction

When a student (or worker) receives a certain task, her subjective estimate of the difficulty of that task has a strong influence on her performance. Research has shown that perceptions of task difficulty are strongly correlated to both performance metrics such as the success rate and the solution time and psychological factors that influence performance such as motivation, interest, self-efficacy and subjective task value (attainment value, intrinsic value, utility value).

Most commonly, the student's perception of the difficulty of a given task is obtained *implicitly* from the description of the task, the background or setting in which the task was provided, the duration of time allotted for the task, or other information related to the task. Yet, there also exist tasks, for which the objective (true) difficulty of the task may be available for a teacher. In this case, the teacher will face an important question: should she

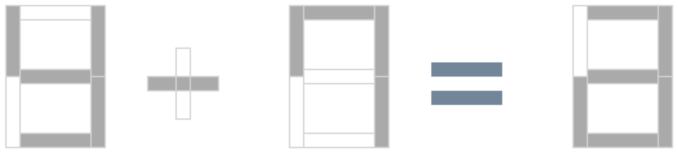


Figure 6.1: Example of a matchstick riddle where 2 matches need to be moved, in order to result in a legal mathematical equation.

reveal the objective difficulty of the task to the student? Will the revelation of the objective task difficulty benefit the student or will it be harmful to her performance?

The influence of perceived task difficulty on the student's performance has been studied extensively [22]. Yet, to the best of our knowledge, the costs and benefits of revealing the objective (true) task difficulty to the student have not been analyzed.

This chapter (which mostly based on [65]) investigates the costs and benefits of revealing the true difficulty of a given task to the student that is supposed to solve that task. In particular, it examines the influence of revealing the task difficulty on performance metrics as well as psychological factors that impact performance, such as motivation, self-efficacy and subjective task value. We study these effects using the matchstick riddle task - a simple mathematical equation in which the numbers and the operator consist of matchsticks, some of which need to be moved in order to obtain a correct mathematical expression (see Figure 6.1 for an example).

For many tasks, the task difficulty is unknown a priori. However, for a given task category (such as matchstick riddles), it might be possible to build a data-driven AI that predicts the difficulty level of a given task from that category. Whether the development of such an AI is beneficial depends on the influence of revealing task difficulty on the student's performance.

Therefore, understanding the influence of revealing task difficulty (in case that it is available) is particularly relevant with respect to AI-supported education.

Whereas the influence of revealing the task difficulty on the student's performance might be consistent (always positive or always negative), it might also be more complex. In particular, it might turn out that this influence depends on several external parameters, such as certain specific attributes of the task or the student's personality. For example, a student who enjoys challenges might become more motivated when she is told that the task is hard while revealing the same information to an anxious student might have the opposite effect. Therefore, in light of these considerations, it might be useful to consider an additional type of AI - an AI that predicts when to reveal the task difficulty and when not.

In summary, this chapter investigates the following questions:

1. What is the influence of revealing the task difficulty on the student's performance, motivation, self-efficacy and subjective task value?

2. How to improve the learning experience using 2 types of AI systems - an AI system that predicts the task difficulty and an AI system that decides when to reveal the task difficulty?

6.2 Related Work

As noted previously, to the best of our knowledge, there is no literature on the influence of revealing the difficulty of a given task on the student's performance. However, the topic of the impact of revealing the task difficulty is closely related to the topic of the impact of perceptions of task difficulty - a subject that has been discussed in a number of scientific publications. This section lists the central conclusions with respect to the influence of perceived task difficulty on performance, on the main psychological factors that influence performance (motivation and self-efficacy), and on subjective task value. One remarkable conclusion from the literature that has been reviewed is the existence of contradicting hypotheses - particularly, concerning the impact of perceived task difficulty on performance and motivation. Since the various studies have taken place on different tasks, one possible explanation for the contradictions might be that the relationship between perceived task difficulty and performance/motivation depends on characteristics that define the specific nature of the given task. The following discussion of the relevant literature is structured in accordance with the following four factors: performance, motivation, self-efficacy and subjective task value.

Performance: In this chapter, the performance of a cohort of students on a task is being measured by two metrics: the success rate and the average solution time. There exists a rich literature on the influence of perceived task difficulty on performance [22]. There are two main hypotheses that appear in the literature. The first hypothesis states that the student's performance decreases as perceived task difficulty increases. According to the second hypothesis, however, the perceived task difficulty has no influence on the student's performance. Although there seems to be more support for the first hypothesis ([22, 43, 38]), there is also some support for the second hypothesis [29]. Although, on the first glance, these two hypotheses seem to contradict each other, it might be the case that upon deeper reflection there is no contradiction between them, because the studies were performed on different tasks and the relationship between perceived task difficulty and performance/motivation might depend on the specifics of the task.

Motivation: The literature on the influence of perceived task difficulty on motivation also contains contradicting hypotheses. The first hypothesis, namely, that motivation is negatively correlated to perceived task difficulty is supported by Hom [29]. Barron et. al. also provide evidence for this hypothesis and augment it with an additional claim; according to their analysis motivation is mediated by self-efficacy. They suggest that an increase in perceived task difficulty leads to a decrease in self-efficacy which causes a decrease in motivation [10]. The contradicting hypothesis, that higher perceived task difficulty leads to higher motivation - appears in the literature as well. Boggiano et. al. suggest that not only the level of motivation but also the level of interest increases when students perceive a task as being more difficult [12]. If high levels of exertion are considered to be an expression of high levels of motivation then the finding that higher perceived task difficulty leads to higher levels of exertion [45] can also be regarded as a support for the contradicting hypothesis. Another alternative hypothesis that can be found in the literature is that there is no relation between perceived task difficulty and motivation. Evidence for this alternative claim can be found in Robinson et. al. [51].

Self-Efficacy: The term "self-efficacy" denotes the student's confidence in her ability to succeed in a given task. There exists a rich literature on the connection between perceived task difficulty, self-efficacy, motivation and performance (see for example [38, 39, 41]). Particularly, it is worth noting, that self-efficacy is closely related to motivation, such that higher self-efficacy usually leads to higher motivation. Therefore, it makes sense to assume that the relationship between perceived task difficulty and self-efficacy is very similar to the relationship between perceived task difficulty and motivation. Since the first hypothesis from the paragraph above - that higher perceived task difficulty leads to lower motivation - has the strongest support in the literature, it might be reasonable to suspect that there also exists literature that confirms the analogous statement with respect to self-efficacy - and, indeed, this seems to be the case (see [78, 79, 80]). Concerning the relationship between high perceived task difficulty and low self-efficacy, it is particularly interesting that it is not clear which factor is the cause and which - the effect. While Wigfield and Eccles suggest that higher perceived task difficulty leads to lower self-efficacy [78], Li et. al. [38] claim the reverse cause-effect relationship: a student is inclined to consider the task as more difficult precisely because she has low confidence in her ability to solve the task.

Subjective Task Value: The term "subjective task value" relates to the various ways in which a given task can be valuable to the student. Expectancy-value theory of motivation [78] defines three types of subjective task values:

- Intrinsic value: the level of pleasure that the student derives from the task.
- Identity value: the amount of self-confidence gained from succeeding in the task.
- Utility value: the value of the knowledge and skills obtained from performing the task.

A review of the literature revealed that perceived task difficulty influences all categories of subjective task value. With regard to intrinsic value, Li et. al. suggest that a student who perceives a task as very difficult will experience lower levels of pleasure in comparison to a student who perceives the same task as less difficult [38]. With regard to identity value, the literature contains multiple hypotheses. While Brown [14] presents evidence for a positive correlation between perceived task difficulty and identity value, Li et. al. [38] do not find any correlation at all. With regard to utility value, we found only one hypothesis in the literature: Li et. al. suggest that utility value increases as perceived task difficulty increases [38].

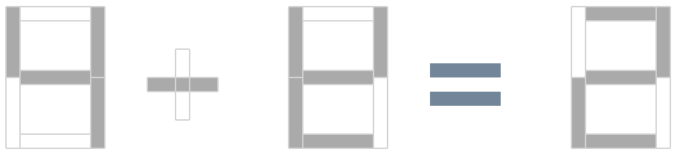


Figure 6.2: Example of a matchstick riddle where 1 match needs to be moved.

6.3 Experimental Setting

6.3.1 The Task

In order to determine the effects of revealing the difficulty of a task to a student we use the matchstick riddle as our test-bed. A matchstick riddle is an equation with 4 tokens (3 digits, 1 operator) and an "=" sign, in which each token consists of multiple matchsticks. The digit tokens can represent any digit 0,1..,9 and the operator token can be a '+' or '-'. The riddle starts with an incorrect mathematical expression, such as '3+2=7' or an expression that contains invalid tokens. In order to solve the riddle, the participant needs to move 1 or more matchsticks to transform the expression into a correct mathematical equation. The amount of matches to be moved is displayed to the participant. Clearly, matchstick riddles have different difficulty levels, see Figure 6.1 for an example of a difficult riddle, and Figure 6.2 for an example of an easy riddle. The solution to Figure 6.1 is "$9 - 1 = 8$", and the solution to Figure 6.2 is "$4 - 2 = 2$".

We use the domain of matchstick riddles because the computer can automatically generate many tasks of different difficulty and easily check if the answer is correct. Furthermore, since this type of task is not well known (unlike mathematical questions, English questions or sudoku), we could expect the performance to be somewhat consistent between different participants, so using the performance of some participants to predict the difficulty of the task may seem more appropriate when using the matchstick riddle.

All experiments (described below) were performed on a collection of matchstick riddles that was generated by a program that we implemented. The collection consists of 500 riddles with 1 match to be moved and another 500 riddles with 2 matches to be moved. Although, as one might intuit, 2-stick riddles are more difficult than 1-stick riddles on average, it turned out that there were many 1-stick riddles that were more difficult than 2-stick riddles, according to the performance statistics. This observation suggests that the difficulty of a riddle depends not only on the number of matches that need to be moved but also on the specific locations of these matches.

6.3.2 Experiment Details

To investigate the influence of revealing task difficulty we conducted an experiment in which the participants (or workers) were asked to solve matchstick riddles. The experiment, which was implemented as a HIT (human intelligence task) on the Amazon Mechanical Turk platform, can be described by the following characteristics.

- The worker saw the riddle and was informed about the number of matches that needed to be moved. To solve the riddle she needed to select the correct matches and to move them into the appropriate positions.

- The order in which the riddles were being displayed to the worker was random (without repetition).

- In order to provide some motivation for the worker, the worker received a small monetary reward for each correct solution.

- The worker had an unlimited amount of time for each riddle. She could move on to the next riddle whenever she liked.
- The worker decided by herself how many riddles she wanted to complete. She was allowed to finish the HIT whenever she desired.
- The worker's performance was measured on each riddle. More precisely, the system measured two variables: the time spent on the riddle and whether the worker was able to solve the riddle or not.

After completing the HIT the worker was asked to fill out a short survey. The survey contained the following statements:

1. I enjoyed the task. (S1)
2. I found the task interesting. (S2)
3. I believe that I performed well on the task. (S3)
4. I would like to perform similar tasks in the future. (S4)

All statements had to be evaluated on a 7-level Likert scale, by choosing an answer from the following list: "strongly disagree" (1), "disagree" (2), "somewhat disagree" (3),"neutral" (4),"somewhat agree" (5), "agree" (6), and "strongly agree" (7).

The experiment consisted of two versions of the riddles HIT. In the first version, the riddle difficulty level was not displayed. In the second version, the difficulty level of the riddle was displayed on top of the riddle. In the following elaboration, the group of workers that performed the 1st/2nd version will be called group 1 and group 2 respectively. The group sizes were N1=97 and N2=125.

For each of the two groups the individual performance statistics were aggregated into group-level performance statistics by averaging. The following group-level statistics have been computed:

- Average attempt ratio, which equals the number of riddles attempted divided by the total number of riddles seen. A riddle is considered to be 'attempted' if the worker spent at least 20s on it.
- Average solution ratio, which is the number of riddles solved divided by the number of riddles seen.
- Average time spent on the HIT.
- Average solution time, which includes only solved riddles.
- Average riddle time, which includes all riddles (solved and unsolved).

All of these group-level statistics were computed not only on the collection of all riddles (table 6.1), but also for the following two subsets of riddles: easy riddles (table 6.2) and hard riddles (table 6.3), as described hereunder. Furthermore, group-level statistics were computed for the survey results.

6.3.3 Determining the Task Difficulty

As mentioned previously, for group 2 the difficulty level of the riddle was displayed on top of each riddle. The possible difficulty levels were 1,2,..,5 where 1 corresponded to 'very easy' and

5 to ‘very hard’. In the discussion of the results the scale that will be used will be a coarser scale with only 3 levels - for greater clarity. The category ‘hard’ will correspond to levels 4 and 5 on the finer scale, the category ‘medium’ will correspond to level 3 and the category ‘easy’ will correspond to levels 1 and 2.

The difficulty level of each riddle was specified before the experiment by running the HIT on a separate group of workers that did not participate in the experiment afterward. The difficulty level of a riddle was determined by the average solution time of that riddle: the higher the average solution time the higher the difficulty level. The size of the group was determined such that each riddle was solved by approximately 10 workers. The function for mapping the average solution time to difficulty level assumes that the number of riddles in each difficulty category is roughly the same. That is, the average solution times are partitioned into five quantiles—one for each difficulty level, where the first quantile contains the lowest solution times (the easiest tasks, i.e., difficulty level 1) and the fifth quantile contains the highest solution times (difficulty level 5).

6.4 Experimental Results

The purpose of the experiment was to understand the influence of revealing task difficulty on multiple performance-related features. This section compares the results of the two groups of workers (group 1 and group 2) with respect to performance and three performance-related features - motivation, self-efficacy and subjective task value. All statistics were computed on the entire collection of riddles and on two subsets: easy riddles and hard riddles. This means that each statistic comes in three versions. For example, there exists an average solution time

	Group 2 (difficulty displayed)	Group 1 (w/o difficulty)
No. workers	129	100
No. riddles seen	4459	3443
Avg seen	34.57	34.43
Avg solved	30.9	30.28
Avg attempted	31.95	31.89
Avg solution ratio	0.89	0.88
Avg attempt ratio	0.92	0.93
Avg time on HIT	3238	2745
Avg solution time	91.92	77.61
Avg riddle time	93.68	79.74

Table 6.1: Performance statistics on the entire riddle collection.

on the set of all riddles, an average solution time on the set of easy riddles and an average solution time on the set of hard riddles.

6.4.1 Influence on Performance

One of the key goals of this chapter is to understand the impact of revealing the task difficulty on the student's performance. The 2 variables that we use to measure the performance, are the solution ratio and the average solution time. We are particularly interested in two questions: (a) Whether revealing that an easy task is easy improves the performance. (b) Whether revealing that a hard task is hard diminishes the performance. For this purpose,

	Group 2 (difficulty displayed)	Group 1 (w/o difficulty)
No. workers	114	89
No. riddles seen	1796	1418
Avg seen	15.75	15.93
Avg solved	14.97	14.75
Avg attempted	15.19	15.13
Avg solution ratio	0.95	0.93
Avg attempted ratio	0.96	0.95
Avg time on HIT	1014	892
Avg solution time	61.18	53.92
Avg riddle time	64.39	56.21

Table 6.2: Performance statistics on easy riddles.

	Group 2 (difficulty displayed)	Group 1 (w/o difficulty)
No. workers	118	86
No. riddles seen	1756	1322
Avg seen	14.88	15.37
Avg solved	12.54	12.69
Avg attempted	13.11	13.67
Avg solution ratio	0.84	0.83
Avg attempted ratio	0.88	0.89
Avg time on HIT	1845	1623
Avg solution time	127.11	106.84
Avg riddle time	124.05	105.6

Table 6.3: Performance statistics on hard riddles.

	Group 2 (difficulty displayed)	Group 1 (w/o difficulty)
No. workers	125	97
Enjoyment	6.34	6.38
Interesting	6.42	6.39
Perform well	5.86	5.81
Similar in future	6.53	6.54

Table 6.4: Survey results

the performance statistics are measured and analyzed both on the entire set of riddles and on the two subsets of riddles (i.e., easy riddles and hard riddles).

The analysis of the experiment results shows that there is no statistically significant difference between the two groups with regard to the solution ratio. This is true both for the entire collection of riddles and for the two subsets. The solution for group 1 and group 2 ratios are similar. For example, on the subset of easy riddles, the average solution ratio is 0.93 for group 1 and 0.95 for group 2.

However, with regard to solution time, it seems that the influence of revealing task difficulty is negative. On the subset of easy riddles, there is no significant difference between the groups. On the subset of hard riddles, however, group 1 clearly outperforms group 2, with 107 seconds for group 1, and 126 seconds for group 2. This result contradicts the initial hypothesis that expecting a task to be easy improves the performance, at least on the matchstick riddle task.

6.4.2 Influence on Motivation

Another important question is the influence of revealing task difficulty on motivation. We use the average attempt ratio and the average riddle time as proxies for the worker's level of motivation. This approach is based on the intuition, that a worker who is more motivated will give up less quickly, which leads to a higher average riddle time as well as to a higher attempt ratio.

The experiment results reveal that on the subset of hard riddles the average riddle time is higher for group 2 than for group 1 (106s (group 1), 124s (group 2)). This result holds for hard riddles only. On the entire collection of riddles and on the subset of easy riddles there is no significant difference in the average riddle time between the 2 groups. This indicates that in certain cases, such as the specific matchstick riddles task, knowing that a task is hard might increase the student's motivation and persistence.

With regard to the attempt ratio, there is no statistically significant difference between the two groups. This result holds both for the entire collection of riddles and each of the two subsets of riddles (easy riddles, hard riddles).

With regard to the survey statistics, the one statement from the survey that might be particularly relevant with respect to motivation is S2 ("I found the task interesting"), as interest facilitates motivation. The experiment results seem to indicate that those workers who were informed about the task difficulty did not find the task more interesting than those workers that were not. There is no significant difference between the 2 groups with regard to the score on S2 with an average of 6.39 for group 1, and 6.42 for group 2.

6.4.3 Influence on Self-Efficacy

The term 'self-efficacy' denotes a student's confidence in his or her ability to succeed in a given task. As noted above, there might be a close relationship between self-efficacy and motivation, such that, in most cases, a student who is highly motivated on a given task has high confidence that she will be able to succeed in it. Therefore, it might be assumed that the influence of revealing task difficulty on self-efficacy is very similar to the influence of revealing task difficulty on motivation.

While it is possible to measure motivation via performance metrics, such as the average attempt ratio, it is not possible to measure self-efficacy this way. Usually, the level of self-efficacy is measured by asking the student about her confidence in succeeding on the task before she starts the task. In our case, however, such an approach would have been infeasible, since in one HIT an average worker attempts about 30 riddles and it might cause her too much discomfort to ask about her confidence level before each riddle. For this reason, we use a different approach. Instead of measuring self-efficacy, we measure the a-posteriori version of self-efficacy: the worker's confidence about her performance on the task after completion. For this purpose, the statement S3 ("Do you believe that you performed well on the task?") was included in the survey that was filled out by every worker after completing the HIT.

The analysis of the survey results (in particular the S3 statement) shows that indeed, the relation between revealing the task difficulty and self-efficacy is very similar to the relation between revealing the task difficulty and motivation. There is no significant difference between the 2 groups in the score on S3, with 5.81 for group 1, and 5.86 for group 2. This result is very similar to the corresponding result with respect to motivation. As discussed previously, those metrics that are indicative of the level of motivation (with the exception of riddle time for hard riddles) were not influenced at all by the revelation of task difficulty.

6.4.4 Influence on subjective Task Value

As mentioned, there are three types of subjective task values: intrinsic value, identity value and utility value. Although the statements in the survey do not precisely correspond to these three types of subjective task value, three of the four statements can be closely linked to them. In particular, it seems meaningful to associate intrinsic value with S1 ("Did you enjoy the task?"), while identity value and utility value might be associated with S2, S4, albeit more loosely.

In order to determine whether revealing task difficulty influences any of the types of subjective task values, the average scores on the corresponding survey statements have been analyzed. With respect to S1, the difference in the scores of 6.38 for group 1 and 6.34 for group 2, is not large enough to be statistically significant. The same conclusion holds true on S2 and S4 as well, where the scores of the two groups are also close to each other. These results reject the intuitive hypothesis that students who are informed about the task difficulty might assign a higher identity value to the task (for example, students might find it rewarding to solve a task when they know that it is hard).

6.5 AI that predicts the Task Difficulty

In the previous section, we have analyzed the influence of revealing task difficulty on the student's performance and some performance-related features such as motivation and self-efficacy. The analysis showed that, indeed, there are certain performance-related features that are positively influenced by revealing the task difficulty, such as average riddle time on hard riddles. In order to reveal the task difficulty to the student, the system (software) that feeds the tasks to the student needs to know the difficulty of every task from the given category of tasks (e.g. matchstick riddles). Although for certain task instances the difficulty level might be known a priori from performance statistics, such as solution time and solution ratio, there may be many other task instances whose difficulty level is unknown. This calls for a difficulty predictor (an AI model) that would be able to compute the difficulty level of any task from a given task category automatically.

Although it might be unrealistic to design a generic difficulty predictor that would operate on all task categories, it might be possible to create one that is specific to a given category of tasks, such as matchstick riddles. For any given category of tasks, this difficulty predictor would operate on a set of features that are specific to that task category. The difficulty predictor can be a model from an arbitrary class of predictor models such as support vector machines or deep neural networks. In case that the chosen class of predictor models is a neural network, the architecture of the network should be chosen with regard to the size of the training set, taking into account the general rule that larger networks require more training data.

In order to train the difficulty predictor, a training set of tasks needs to be generated. This can be accomplished in a similar way as was done for the matchstick riddles collection (see "Experimental setting" section). The first step would be the generation of a collection of tasks from the given category. Afterward, a cohort of workers should be asked to solve these tasks and certain performance statistics such as the solution times and the solution ratios should be recorded. In the final step, these performance statistics might be leveraged to determine the difficulty level of each task in the training set.

For many task categories predicting the difficulty level of a task might be much simpler than solving it since the former only requires training a difficulty predictor on task-difficulty-tuples, while the latter calls for the development of a solution algorithm for the given task category, which might be extremely hard or even impossible for certain task categories. Therefore, it is important to emphasize that the task difficulty predictor does not need to know the solution of the task. This is a very important feature of the predictor because it allows to build a predictor even when a solution algorithm is not available.

6.6 Predicting Task Difficulty on the Matchstick Riddles Task

To test the ideas presented in the previous section we built a solution time predictor for the matchstick riddles task. Since the difficulty of a riddle, as we defined it, was determined only by the solution time, the solution time prediction can be transformed into a difficulty prediction. Because of the staggering success of deep learning in various domains in recent years, the function class that was chosen for the predictor was the class of fully connected neural networks with rectified linear unit (ReLU) neurons. In order to obtain optimal performance, multiple network architectures were tested - all of them consisting of 2-4 hidden layers and 10-20 neurons per layer. Furthermore, each layer was regularized by a dropout layer with a dropout rate of 0.5. The model training was performed using an early stopping strategy (the training session ended when the validation error did not decrease for a period of 10 consecutive epochs).

The basis for the data, that was used for training, validating and testing the model was a collection of 1300 matchstick riddles and the corresponding solution times. The experiment yielded 5-10 solution times for every riddle (each solution time corresponds to a different worker). The median of these solution times was chosen as the prediction target. Thus, each sample in the resulting data set consisted of the riddle's feature vector and the corresponding median solution time. We chose to predict the median solution time rather than the average solution time, because the median is more robust to outliers than the average. Finally, the data was partitioned into a training set (60 %), a validation set (20 %) and a testing set (20 %).

In order to understand how the choice of features influences the quality of the difficulty predictor, multiple difficulty predictors have been trained - each one on a different set of features. The first predictor leveraged a minimal collection of features that used only the information contained in the riddle, but not the solution. This feature collection consisted of the match positions of the original expression (encoded as a one-hot vector) and the number of matches that needed to be moved. The second predictor was based on an extended feature collection that utilized the task solution. This extended collection consisted of the minimal feature collection, augmented by a one-hot vector, that encoded the initial and final positions of those matches that needed to be moved and a boolean feature that indicates whether there is a change in the operator ('+' to '-' or '-' to '+').

After training, all models were evaluated on the test set. With regard to the model performance, the differences between the different model architectures that were tested were negligible. Moreover, it was observed that adding the solution-based features to the minimal feature collection did not improve the model performance. All models, including those that used the extended feature collection, yielded a mean average error (MAE) of approximately 35 seconds on the test set. To evaluate the predictive power of the models, they were compared to a primitive model that produces a constant output for every riddle: the average target value (averaged over the training set). This primitive model produced a MAE of 42 seconds. Although this result is worse than the MAE of the neural network, unfortunately, the differences are quite small.

There might be multiple explanations for the relatively poor performance of the neural network solution time predictor on the matchstick riddles task. One possible explanation for this phenomenon is the complexity of the given prediction task. It is not clear which features determine the complexity - aside from the number of matches that need to be moved. Initially, we hypothesized that 2 features that are likely to make a riddle more difficult are a change in the operator and a change in the right side. Yet, the analysis of the data does not support this hypothesis. Furthermore, it might be worth noting that the prediction task (predicting the median solution time) cannot be accomplished by a human within a few seconds - a fact that might be another indicator of the complexity of the prediction task.

6.7 AI that decides when to reveal the Task Difficulty

The analysis of the experiment results showed, that revealing task difficulty indeed does have an impact on the student's performance and motivation, albeit only on certain subsets of tasks (for example on hard riddles). Yet, from the experiment results, it did not become clear whether this influence is positive or negative. As reviewed above, the literature on the influence of perceived task difficulty contains contradicting hypotheses. One possible interpretation of this finding might be the hypothesis that this influence might depend on additional factors that were not taken into account, such as certain specifics of the task or the student's attributes. For instance, an anxious student might easily lose motivation by being told that the task is hard, while a very self-confident student, in contrast, might gain additional motivation from this information.

The assumption that the influence of revealing task difficulty depends on external factors, such as specific of the given task category or the student's personality, calls for an AI system that would determine in which cases task difficulty should be revealed. The inputs for this system might include various features, such as scores for the student's personality traits and possibly certain features of the given task category. The system's output could be either binary ('yes' or 'no') or continuous (a score that indicates the benefit of revealing the task difficulty). A more sophisticated system could produce scores that would reflect how beneficial revealing task difficulty would be with regard to every single performance-related feature: performance, motivation, self-efficacy and subjective task value. It should be mentioned that at this point the description of this system is rather abstract because more research would be required to understand how to design such a system. In particular, more research would be required to understand which factors modulate the influence of revealing task difficulty, how exactly they modulate this influence and how they correlated with each other.

There exists evidence that confirms that a student's level of motivation on a given task depends on her personality [44]. Therefore, the student's personality profile could be a key factor that determines whether revealing the task difficulty has a positive or negative impact on her motivation, which means that a model that determines in which cases task difficulty should be revealed, might require the student's psychological profile. This psychological profile should contain scores on various personality traits, such as trait neuroticism and trait openness, which might be relevant with regard to the influence of perceived task difficulty. The field of psychometrics offers numerous ways to generate such a personality profile, among others, specialized psychological tests and questionnaires.

6.8 Discussion & Conclusion

The current chapter explored the influence of revealing the difficulty level of a task on the student's performance and several performance-related features such as motivation, self-efficacy and subjective task value. The purpose of this effort was to understand whether revealing task difficulty improves the student's performance and to discuss how to support a student with two types of AI systems: an AI that predicts the task difficulty and an AI that determines when to reveal the task difficulty. The matchstick riddles experiment produced several interesting results. While some of these results are intuitive and in line with the existing literature, others are less expected.

Three important experiment results, which are in line with the literature, can be regarded as intuitive. The first result is related to the solution time, which is an indicator of performance. It has been observed that the solution time increased when the worker was told that the riddle is hard. This result is in line with the hypothesis from [22, 43, 38] that states that an increase in perceived task difficulty causes a decrease in performance. The two other notable results are linked to the influence of revealing task difficulty on motivation and interest. With regard to these two features (motivation and interest), the experiment results indicate that revealing the difficulty of the riddles did not influence neither motivation nor interest.

Another result, related to performance, contradicts certain findings from the literature [22, 43, 38]. Namely, when the worker was informed that a task is easy, her performance did not improve - neither with respect to the solution time nor with respect to the solution ratio.

With regard to motivation, there are 2 counterintuitive experiment results that are also in contradiction to certain results from the literature [51] . Firstly, it was observed that the worker's motivation did not decrease when she was informed that a riddle is hard—there was even evidence for an increase in motivation. Secondly, when a worker was informed that a riddle is easy, her motivation did not increase.

In conclusion, the riddles experiment showed that the impact of revealing task difficulty is neither clearly positive nor clearly negative, but depends on different factors such as the specific performance-related feature that is being inspected (performance, motivation etc.) and the difficulty of the task. We believe that there are four major conclusions that can be drawn from the experiment:

- **Conclusion 1**: Revealing task difficulty has positive as well as negative effects on the student's performance and performance-related features (motivation etc.).
- **Conclusion 2**: In some cases, the impact of revealing task difficulty can be ambiguous (positive and negative simultaneously). For example, revealing task difficulty on hard tasks leads to an increase in solution time, which can be regarded either as a decrease in performance or as an increase in exertion.
- **Conclusion 3**: Revealing the task difficulty seems to impact only some of the performance-related features (performance and motivation) but has no impact on others (self-efficacy, subjective task value).
- **Conclusion 4**: The impact of revealing task difficulty might depend on the specifics of the task and the student's personality.

It is important to emphasize that the experiment discussed in this chapter was performed on a single task category (matchstick riddles). Thus, it is not certain that the conclusions, derived from the experiment results can be generalized to other task categories. For this reason, an interesting direction of research for future work could be to perform a similar experiment on other task categories (e.g. sudoku).

The goal of this chapter was to investigate the utility of supporting students with an AI that predicts task difficulty for a given task category (such as matchstick riddles). The analysis of the experiment results showed that in many cases revealing task difficulty can have positive effects. Therefore, an AI that predicts task difficulty might be useful for increasing the students' performance and motivation. On the other hand, the experiment results also indicate that sometimes revealing task difficulty impacts the student in a negative way. For this reason, the difficulty predictor should be augmented with an AI that determines when to reveal task difficulty and when not to do so.

Chapter 7

Conclusion

This thesis introduced the concept of criticality in AI. Briefly, the criticality of an environment state is the impact of the action choice in this state on the agent's performance, while the criticality of an action (in particular in the context of AI safety) indicates the level of potential harm that can result from that action. The main motivation for introducing this novel concept came from the observation that people adjust their mode of operation to the criticality level of the situations that they encounter and the question of whether such an adjustment would be beneficial for AI agents too. To investigate this question the thesis proposed multiple learning algorithms that utilize state criticality information and analyzed their performance in multiple environments. The experiments that I presented, showed that it is possible to improve learning algorithms by incorporating criticality information. In addition, the thesis also included a chapter on the application of task difficulty information in education. Although task difficulty is not exactly the same as state criticality, this metric can be regarded as a form of criticality in a broader sense since critical tasks are often difficult and difficult tasks can be often critical from an educational perspective.

The thesis presented 4 applications of the novel concept of criticality in AI. The first application is the CVS algorithm which lives in the domain of reinforcement learning. CVS is a n-step learning algorithm with a flexible stepnumber that depends on the local state criticality. The second application of criticality to AI – the CBA algorithm – belongs to the domain of advice-based reinforcement learning. In CBA criticality information is leveraged to decide which states to select for advice. The third application of criticality, discussed in the thesis, is related to the field of AI safety. In this application, the action criticality metric is used to distinguish risky instructions from safe ones in order to reduce the workload of a human safety agent monitoring an AI agent. The last of the four applications of criticality is, strictly speaking, more related to task difficulty than to criticality. In that particular piece of research, I discuss the question of whether students might have a better learning experience if they are informed about the difficulty levels of the tasks that they need to solve. To investigate this question, I designed a study that measured the impact of receiving task difficulty information on the students learning experience. This study particularly is relevant in the context of educational AI because it helps to evaluate whether it is worth investing resources in developing AI systems that predict task difficulty levels for various task categories.

The first application of state criticality discussed in this thesis is the CVS algorithm whose purpose is to circumvent the problem of finding an appropriate stepnumber in n-step learning algorithms for reinforcement learning. The central idea of this algorithm is to use a flexible stepnumber that depends on local state criticality instead of choosing a fixed stepnumber like in the ordinary n-step algorithm. In order to analyze the algorithm's efficiency, the CVS algorithm was validated in 3 learning environments. The experiment results showed, that in every environment CVS was able to outperform its competitors. Research on CVS can be extended in multiple directions. One viable research direction might consist of generalizing CVS to learning algorithms that use eligibility traces, such as Q(λ). It might be also interesting to examine the possibility of learning the criticality function from a training set of state/criticality tuples generated by a human, instead of using a predefined criticality function. This research direction might be relevant to all criticality-based methods since in complex environments it is impossible to ask the human for a criticality function that covers every environment state.

The second application of criticality that was introduced in this thesis was the CBA algorithm, which addresses one of the central challenges in the domain of advice-based RL: efficient selection of advice states. The CBA algorithm utilizes criticality as a criterion for advice state selection, such that the probability for receiving advice in a given state is proportional to the state's criticality. To analyze the algorithm's efficiency experiments were performed in 2 environments. The experiment results indicated that state criticality can be used to improve advice state selection in advice-based RL. There are multiple interesting research directions related to the use of criticality in advice-based RL. Among others, it might be interesting to explore how CBA performs in environments with a smaller portion of highly critical states than in the test environments used in the experiments. Another potential research direction could consist of replacing static criticality with skill-dependent criticality to prevent the agent from asking for advice in states where the proper action was learned already.

The third application of criticality in AI that was introduced in this thesis is related to the AI safety domain. Often, AI agents that operate in the real world can cause real damage and therefore need to be supervised by a human safety agent. Although supervision reduces the AI risk, the workload for the human safety agent might be quite high, in particular, if the operating AI agent frequently formulates action instructions (for himself). To reduce the workload for the human safety agent, I suggested introducing an AI safety system that would distinguish potentially harmful instructions from (almost) safe ones by applying linguistic analysis. The suggested method for the linguistic analysis uses 3 distinct criticality metrics: object-based-, value-based-, and verb-based criticality. Object-based criticality measures the potential danger related to objects and substances. Value-based criticality identifies instructions that include particularly valuable objects. Verb-based criticality is used to identify instructions that contain risky actions. In the thesis, I describe different ways to combine these 3 criticality measures in order to determine the total criticality of an instruction. Although the suggested system, indeed, might be able to make risk management for AI agents more efficient, a substantial amount of research might be required to design a safety filter that would be both safe and efficient (identifies a sufficiently high percentage of risky as well as almost safe instructions correctly) – especially for complex environments.

The last application of criticality discussed in this thesis is connected educational AI. The question that motivated that particular piece of research was whether a student who is supposed to solve a given task will benefit from receiving information about the difficulty of that task. Although this question seems to be unrelated to AI, indeed, there is a connection since understanding the impact of revealing the task difficulty is important for evaluating the utility of AI systems that predict task difficulty. Using a matchstick task, we explored the influence of task difficulty revelation on the student's performance, motivation, self-efficacy and subjective task value. The study indicated that task difficulty revelation has some limited impact on performance and motivation (only on hard tasks) and has no impact on self-efficacy and subjective task value. Since the study did not account for the differences in the student's personalities it might be interesting to examine how the impact of task difficulty revelation is modulated by the student's character traits. For example, an anxious student might become scared from a difficult task whereas a student who enjoys challenges might gain additional motivation if he is informed that the task is rather difficult.

As mentioned previously, in this thesis I presented 4 ways of using the novel concept of state criticality to improve the learning and safety of AI systems, and, furthermore, I sketched 2 other applications of criticality: one related to skill preservation – the other to education. The experiments presented, in particular, on the 2 applications of criticality to RL, demonstrated the power of criticality-aware learning algorithms. Although integration of criticality into various AI algorithms may improve their efficiency, there is also a price to pay. The main drawback of criticality-aware learning algorithms is their need for an appropriate criticality function. The experiments presented in the thesis were performed in relatively simple environments where it was rather easy to come up with a meaningful criticality function. Yet, in complex environments it might be much harder (or even impossible) for a human, to generate a criticality function that approximates the true criticality of all states in the given environment. Hence, it is necessary to find ways of generating a criticality function that do not require the human to define this function manually. For this purpose, it might be interesting to explore the possibility of learning a criticality function from a training set of state/criticality tuples generated by a human (or a crowd). Since this task itself might be just as challenging as the original problem (solving the MDP) it might be also interesting to consider ways of utilizing an underdefined criticality function, which is defined only on some environment states and not defined on others.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 2004.

[2] David Abel, James MacGlashan, and Michael L. Littman. Reinforcement learning as a framework for ethical decision making. In *AAAI Workshop: AI, Ethics, and Society*, volume WS-16-02 of *AAAI Workshops*. AAAI Press, 2016. 978-1-57735-759-9.

[3] Colin Allen, Iva Smit, and Wendell Wallach. Artificial morality: Top-down, bottom-up, and hybrid approaches. *Ethics and Inf. Technol.*, 7(3), September 2005.

[4] Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara J. Grosz. Interactive teaching strategies for agent training. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 804–811. AAAI Press, 2016.

[5] Thomas Arnold and Matthias Scheutz. Against the moral turing test: accountable design and the moral reasoning of autonomous systems. *Ethics and Information Technology*, 18, 03 2016.

[6] Kristopher De Asis and Richard S. Sutton. Per-decision multi-step temporal difference learning with control variates. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 786–794, 2018.

[7] Amos Azaria, Zinovi Rabinovich, Sarit Kraus, Claudia Goldman, and Ya'akov Gal. Strategic advice provision in repeated human-agent interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[8] Amos Azaria, Zinovi Rabinovich, Sarit Kraus, Claudia V Goldman, and Omer Tsimhoni. Giving advice to people in path selection problems. In *AAMAS*, pages 459–466, 2012.

[9] Amos Azaria, Ariel Rosenfeld, Sarit Kraus, Claudia V Goldman, and Omer Tsimhoni. Advice provision for energy saving in automobile climate-control system. *AI Magazine*, 36(3):61–72, 2015.

[10] Kenneth Barron and Chris Hulleman. Expectancy-value-cost model of motivation. *International Encyclopedia of the Social and Behavioral Sciences*, pages pp. 503–509 (Vol. 8), 01 2014.

[11] Joseph A. Blass and Kenneth D. Forbus. Moral decision-making by analogy: Generalizations versus exemplars. In *AAAI*, pages 501–507. AAAI Press, 2015.

[12] Diane N. Boggiano, Ann K.; Ruble. Competence and the overjustification effect: A developmental study. *Journal of Personality and Social Psychology*, 1979.

[13] Selmer Bringsjord, Konstantine Arkoudas, and Paul Bello. Toward a general logicist methodology for engineering ethically correct robots. *IEEE Intell. Syst.*, 21(4):38–44, 2006.

[14] Carol Brown. The perceived role of task difficulty and effort in the expectations and values of a level students. *Psychology of Education Review*, 2018.

[15] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E. Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3352–3358, 2015.

[16] Nicholas Carr. *The Glass Cage: How Our Computers Are Changing Us.* W.W. Norton Company, 2015.

[17] Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. 09 2017.

[18] William R. Clements, Benoît-Marie Robaglia, Bastien Van Delft, Reda Bahi Slaoui, and Sébastien Toth. Estimating risk and uncertainty in deep reinforcement learning. *CoRR*, abs/1905.09638, 2019.

[19] Francisco Cruz, Johannes Twiefel, Sven Magg, Cornelius Weber, and Stefan Wermter. Interactive reinforcement learning through speech guidance in a domestic scenario. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 1–8. IEEE, 2015.

[20] Felipe Leno Da Silva, Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. Uncertainty-aware action advising for deep reinforcement learning agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5792–5799, Apr. 2020.

[21] Morteza Dehghani, Emmett Tomai, and Matthew Klenk. An integrated reasoning approach to moral decision-making. volume 3, pages 1280–1286, 01 2008.

[22] Scasserra Dominick. The influence of perceived task difficulty on task performance. *https://rdw.rowan.edu/etd/756*, 2008.

[23] Avshalom Elmalech, David Sarne, Esther David, and Chen Hajaj. Extending workers' attention span through dummy events. In *Proceedings of the Fourth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2016, 30 October - 3 November, 2016, Austin, Texas, USA.*, pages 42–51, 2016.

[24] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L. Isbell, and Andrea Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2625–2633, Red Hook, NY, USA, 2013. Curran Associates Inc.

[25] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L. Isbell, and Andrea Lockerd Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2625–2633, 2013.

[26] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, pages 3909–3917, 2016.

[27] Gabriel Hartmann, Zvi Shiller, and Amos Azaria. Deep reinforcement learning for time optimal velocity control using prior knowledge. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 186–193. IEEE, 2019.

[28] Gabriel Hartmann, Zvi Shiller, and Amos Azaria. Model-based reinforcement learning for time-optimal velocity control. *IEEE Robotics and Automation Letters*, 5(4):6185–6192, 2020.

[29] Harry L. Hom. The impact of task difficulty expectations on intrinsic motivation. *Motivation and Emotion*, 1983.

[30] Sandy H. Huang, K. Bhatia, P. Abbeel, and A. Dragan. Establishing appropriate trust via critical states. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3929–3936, 2018.

[31] Ercument Ilhan, Jeremy Gow, and Diego Perez Liebana. Teaching on a budget in multi-agent deep reinforcement learning. pages 1–8, 08 2019.

[32] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. *arxiv:1511.03722*, 2015.

[33] Kshitij Judah, Saikat Roy, Alan Fern, and Thomas G. Dietterich. Reinforcement learning via practice and critique advice. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[34] W. Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *The Fifth International Conference on Knowledge Capture*, September 2009.

[35] W. Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The TAMER framework. In *The Fifth International Conference on Knowledge Capture*, September 2009.

[36] Toby Jia-Jun Li, Amos Azaria, and Brad A Myers. Sugilite: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 6038–6049, 2017.

[37] Toby Jia-Jun Li, Igor Labutov, Brad A Myers, Amos Azaria, Alexander I Rudnicky, and Tom M Mitchell. Teaching agents when they fail: end user development in goal-oriented conversational agents. In *Studies in Conversational UX Design*, pages 119–137. Springer, 2018.

[38] Weidong Li, Amelia Lee, and Melinda Solmon. The role of perceptions of task difficulty in relation to self-perceptions of ability, intrinsic value, attainment value, and performance. *European Physical Education Review*, 13(3):301–318, 2007.

[39] Weidong Li, Amelia Lee, and Melinda Solmon. Effects of dispositional ability conceptions, manipulated learning environments, and intrinsic motivation on persistence and performance: An interaction approach. *Research quarterly for exercise and sport*, 79:51–61, 04 2008.

[40] Patrick Lin, Keith Abney, and George A. Bekey. *Robot Ethics: The Ethical and Social Implications of Robotics*. The MIT Press, 2014.

[41] Phillip Mangos and Debra Steele-Johnson. The role of subjective task complexity in goal orientation, self-efficacy, and performance relations. *Human Performance - HUM PERFORM*, 14:169–185, 04 2001.

[42] Maja J. Mataric. Reward functions for accelerated learning. In *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pages 181–189, 1994.

[43] Douglas C. Maynard and Milton D. Hakel. Effects of objective and subjective task complexity on performance. *Human Performance*, 10(4):303–330, 1997.

[44] Edward McAuley, Terry Duncan, and Vance V. Tammen. Psychometric properties of the intrinsic motivation inventory in a competitive sport setting: A confirmatory factor analysis. *Research Quarterly for Exercise and Sport*, 60(1):48–58, 1989. PMID: 2489825.

[45] Sharma Meenakshi. Effect of perceived goal difficulty, perceived exercise exertion and sub-goal on selected motor task. *academia.edu*, 2019.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[47] Anis Najar and Mohamed Chetouani. Reinforcement learning with human advice. A survey. *CoRR*, abs/2005.11016, 2020.

[48] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[49] Monica Nicolescu and Maja Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. pages 241–248, 01 2003.

[50] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[51] P Robinson. Task complexity, task difficulty, and task production: exploring interactions in a componential framework. *Applied Linguistics*, 22(1):27–57, 2001.

[52] Ariel Rosenfeld, Amos Azaria, Sarit Kraus, Claudia V Goldman, and Omer Tsimhoni. Adaptive advice in automobile climate control systems. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 543–551, 2015.

[53] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 101–103, New York, NY, USA, 1998. Association for Computing Machinery.

[54] Stuart Russell, Daniel Dewey, and Max Tegmark. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4):105–114, Dec. 2015.

[55] Paul E. Rybski, Kevin Yoon, Jeremy Stolarz, and Manuela M Veloso. Interactive robot task training through dialog and demonstration. New York, NY, USA, 2007. Association for Computing Machinery.

[56] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *CoRR*, abs/1704.02532, 2017.

[57] Stefan Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997.

[58] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016.

[59] Felipe Silva and Anna Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64, 03 2019.

[60] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017.

[61] Yitzhak Spielberg and Amos Azaria. The concept of criticality in reinforcement learning. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 251–258. IEEE, 2019.

[62] Yitzhak Spielberg and Amos Azaria. The concept of criticality in reinforcement learning. *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 251–258, 2019.

[63] Yitzhak Spielberg and Amos Azaria. The concept of criticality in ai safety. *arxiv*, 2020.

[64] Yitzhak Spielberg and Amos Azaria. Criticality-based varying step-number algorithm for reinforcement learning. *Int. J. Artif. Intell. Tools*, 30(4):2150019:1–2150019:21, 2021.

[65] Yitzhak Spielberg and Amos Azaria. Revelation of task difficulty in al-aided education. In *33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021, Washington, DC, USA, November 1-3, 2021*, pages 1403–1408. IEEE, 2021.

[66] Yitzhak Spielberg and Amos Azaria. Criticality-based advice in reinforcement learning. *AAAI (student abstract)*, 2022.

[67] Ron Sun. Moral judgment, human motivation, and neural networks. *Cognitive Computation*, 5, 12 2013.

[68] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. 2017.

[69] Richard S. Sutton, Ashique Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of Machine Learning Research*, 17:73:1–73:29, 2016.

[70] Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *(AAMAS 2011), Volume 1-3*, pages 617–624, 2011.

[71] Ana C. Tenorio-Gonzalez, Eduardo F. Morales, and Luis Villaseñor-Pineda. Dynamic reward shaping: Training a robot by voice. In Angel Kuri-Morales and Guillermo R. Simari, editors, *Advances in Artificial Intelligence – IBERAMIA 2010*, pages 483–492, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[72] Sanjay Thakur, H. V. Hoof, J. Higuera, Doina Precup, and D. Meger. Uncertainty aware learning from demonstrations in multiple contexts using bayesian neural networks. *2019 International Conference on Robotics and Automation (ICRA)*, pages 768–774, 2019.

[73] Andrea L. Thomaz and Cynthia Breazeal. Robot learning via socially guided exploration. In *2007 IEEE 6th International Conference on Development and Learning*, pages 82–87, 2007.

[74] Lisa Torrey and Matthew Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. *AAMAS*, 2013.

[75] Lisa Torrey and Matthew Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. *AAMAS*, 2013.

[76] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 30, 2016.

[77] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[78] Wigfield and Eccles. Expectancy-value theory of achievement motivation. *Contemporary educational psychology*, 25 1:68–81, 2000.

[79] Allan Wigfield and Jacquelynne S Eccles. The development of achievement task values: A theoretical analysis. *Developmental Review*, 12(3):265–310, 1992.

[80] Allan Wigfield and Jacquelynne S Eccles. Expectancy-value theory of achievement motivation: A developmental perspective. *Educational Psychology Review*, 6:49–78, 1994.

[81] Wolchover. Concerns of an artificial intelligence pioneer. *Quanta Magazine*, 2015.

[82] Matthieu Zimmer, Paolo Viappiani, and Paul Weng. Teacher-student framework: A reinforcement learning approach. 05 2014.

תקציר

כאשר אנשים פועלים בעולם או עוסקים בתהליך למידה הם נתקלים במצבים שונים. ישנם מצבים שבהם לבחירת הפעולה יש השפעה מינורית בלבד על חייו, למשל, כאשר אדם צריך לבחור בין מספר וריאנטים של סועדים, וישנם מצבים אחרים בהם השפעה זו גדולה בהרבה (מצבים קריטיים), למשל, כאשר אדם צריך לבחור את בן זוגו לחיים. באופן טבעי, אנשים מתאימים את מצבם הפיזי והנפשי לסוג המצב שהם נתקלים בהם ובהתאם לשכל הישר שאומר שמצבים קריטיים דורשים מחשבה אינטנסיבית יותר ובאופן כללי השקעת משאבים גבוהה יותר. ברור שגם סוכני בינה מלאכותית במהלך אימון או עם פריסה נתקלים במצבים עם רמות קריטיות שונות. לפיכך, הנושא המרכזי של תזה זו הוא כיצד סוכני AI יכולים להתאים את אופן הלמידה/פעולה שלהם לרמות הקריטיות של המדינות שבהן הם מבקרים.

בתזה זו, אני מציג מושג חדש בתחום הבינה המלאכותית --- מושג הביקורתיות. מכיוון שקריטיות היא מושג מופשט למדי, במקום לספק הגדרה שמתקיימת עבור כל תחומי הבינה המלאכותית, תזה זו מציעה הגדרות שונות של קריטיות עבור תחומי הבינה המלאכותית השונים. בנוסף להצגה ולדיון כיצד הרעיון החדש יכול לבוא לידי ביטוי בכמה תחומים נבחרים של בינה מלאכותית, המיקוד העיקרי של התזה הוא להראות שניתן להשתמש בו כדי לעזור לסוכני בינה מלאכותית ללמוד מהר יותר ולפעול בצורה בטוחה ויעילה יותר . לצורך כך מציגה התזה 3 יישומים של קריטיות: שניים בהקשר של למידת חיזוק (פרקים 2 ו-3) ואחד בהקשר של בטיחות בינה מלאכותית (פרק 4).

בעוד שארבעת הפרקים הראשונים דנים ביישומים של המושג החדש במספר תחומים של בינה מלאכותית, פרק 5 אינו עוסק בביקורתיות במובן הצר של המונח אלא סובב סביב מדד שניתן להתייחס אליו כקשור הדוק לקריטיות, כלומר, קושי במשימה. . בהקשר של בינה מלאכותית בחינוך, השאלה המרכזית של אותו פרק היא האם גילוי הקושי במשימה משפר את חווית הלמידה של התלמיד. קושי במשימה קשור לביקורתיות משום שהוא מצריך התאמה של אופן הפעולה: משימה קשה מחייבת את התלמיד להפגין מאמץ מנטלי ואינטלקטואלי מרבי. לכן, בהקשר של חינוך, קושי במשימה יכול להיחשב כהיבט של ביקורתיות.

אוניברסיטת אריאל

מושג הקריטיות בבינה מלאכותית

חיבור זה הוגש כחלק מדרישות התואר "דוקטור לפילוסופיה"

מאת

יצחק שפילברג

עבודה זו נכתבה בהנחיית

פרופ' ואדים לויט

פרופ' עמוס עזריה

מוגש לסנאט אוניברסיטת אריאל בשומרון

מרץ 2022