The Concept of Criticality in Reinforcement Learning

Yitzhak Spielberg
Department of Computer Science
Ariel University
Ariel, Israel
yspielb@gmail.com

Amos Azaria
Department of Computer Science
Ariel University
Ariel, Israel
amos.azaria@ariel.ac.il

***Abstract*—This paper introduces a novel idea in human-aided reinforcement learning - the concept of criticality. The criticality of a state indicates how much the choice of action in that particular state influences the expected return. In order to develop an intuition for the concept, we present examples of plausible criticality functions in multiple environments. Furthermore, we formulate a practical application of criticality in reinforcement learning: the criticality-based varying stepnumber algorithm (CVS) - a flexible stepnumber algorithm that utilizes the criticality function, provided by a human, in order to avoid the problem of choosing an appropriate stepnumber in n-step algorithms such as n-step SARSA and n-step Tree Backup. We present experiments in the Atari Pong environment demonstrating that CVS is able to outperform popular learning algorithms such as Deep Q-Learning and Monte Carlo.**

***Index Terms*—Human-aided reinforcement learning; Human-agent interaction**

I. 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 I have for dinner?", "Should I invest two extra hours to work on one's project or spend the evening watching a movie?", "Which route should I 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 do I want to live? ", "Which profession do I want to possess?" , "How much do I want to invest in my health?" "How do I want to educate my 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. There is a variety of ways the expected return (the Q-values) in that particular state can be distributed among the possible actions. To list only a few of them: all actions could have the same Q-value; the Q-values might have an approximately uniform distribution; there might be one or multiple catastrophic actions.

In this paper 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 the human trainer. In such a learning scenario there might be algorithms that use criticality in order to boost the agent's performance. In this paper we present one such learning algorithm: the criticality-based varying stepnumber algorithm (CVS). CVS might be regarded as an algorithm that is closely related to the class of n-step learning algorithms (with a fixed stepnumber), such as n-step SARSA and n-step Tree Backup. However, as a flexible stepnumber algorithm, CVS does not suffer from the central problem of fixed stepnumber algorithms: the problem of choosing an appropriate stepnumber. This paper introduces the concept of criticality and shows its application in the context of reinforcement learning. We believe that criticality input may be used by a reinforcement learning agent in several ways, however, in this paper we propose only one single method for using criticality in reinforcement learning: the Criticality-based Varying Stepnumber algorithm (CVS).

II. The Concept of Criticality in Reinforcement Learning

A. 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 will hit that obstacle, the student will understand that she probably took a wrong action back when the teacher has 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 paper).

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. Yet, the optimal policy should permeate the definition of criticality at least in an informal manner, since it is exactly *that* policy, that the agent is supposed to learn. One way of achieving this would be by asking the human trainer to have in mind a subjective (and maybe merely subconscious) estimate of the optimal policy, during the mental process of determining a criticality level for a given state. Such a requirement would on the one hand be friendly to the human teacher and on the other hand, it would ensure that the criticality measure is meaningful in the context of learning the optimal policy.

B. 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, for example in the Atari Pong environment (which is described in one of the following sections), the workload for the human trainer is rather limited. 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 (maybe the accumulation is not linear since many states are similar to each other), 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. The "experiments" section contains the learning curve of agents that operate according to each of these approaches.

C. 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 (resulting in a negative reward). This implies that the state was in fact a critical 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), but the criticality seems high (since there exists a catastrophic action). 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 requires transforming the weighted variance to a value between 0 and 1 (by some kind of normalization procedure), but may be closer to what humans view as critical states. 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 than for a professional player, who is more likely to score.

III. 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 which 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 (one is the agent and the other is the opponent), a ball, and a playing field. The movements of each racket are defined by the three primitive actions (up, down, stay). 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 probability 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 him.

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).

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.

IV. CVS

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

A. The Relation between Criticality and the Stepnumber

All prominent n-step reinforcement learning algorithms, such as n-step SARSA, n-step Expected SARSA and n-step Tree Backup, use a fixed stepnumber 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 stepnumber that is specific to each state encountered during an episode, and we use criticality to determine the appropriate stepnumber for a given state.

In order to develop some intuition on the way in which criticality could be used to determine an appropriate stepnumber, 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 "choice" 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 him 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 perfectly 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.

B. The 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 which 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 stepnumbers in more critical domains of the state space. We will call this algorithm "Criticality-based Varying Stepnumber" (CVS).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 $Q(S', A')$

V. Evaluation of CVS in the Atari Pong Environment

A. 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 (width x length) 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.

B. 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) [17] 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 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)].

C. 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.

D. 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

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

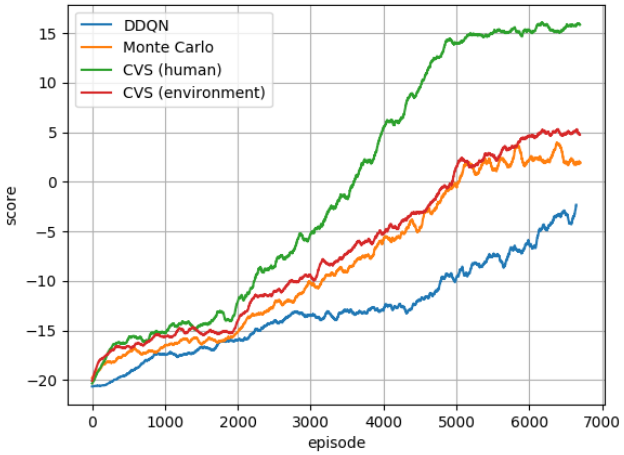

Fig. 1. CVS vs DDQN and Monte Carlo Atari Pong environment: average scores (5 simulations). The CVS(human) agent works with a criticality function that was designed by a human. The CVS(environment) agent learns the criticality from the environment. The CVS(human) agent clearly outperforms both competitors, whereas the CVS(environment) agent performs very similar to the Monte Carlo agent.

E. 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 1. 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 DDQN agent. The Monte Carlo agent performs better than the DDQN agent as well, although not as good as the CVS(human) agent. The CVS(environment) agent's performance level is very similar to that of the Monte Carlo agent.

VI. Discussion

We tested CVS against two algorithms that correspond to the two opposite extremes within the spectrum of the stepnumber parameter in n-step algorithms. The DDQN algorithm is a specific type of Q-Learning algorithm and therefore corresponds to the value $n = 1$; the Monte Carlo algorithm corresponds to the value $n = \infty$. CVS was able to outperform both of them. An interesting question could be, how CVS performs against intermediate values of n. We did not claim that CVS rivals the fixed-n-step algorithm for any level of n but in fact this *might* be the case.

In addition to the human-aided version of CVS, where the criticality is being provided by the human teacher we tested a version of CVS, in which the agent learns criticality from

the environment. We saw that later version of CVS does not perform as good as the former. An analysis of the criticality values that were obtained from the environment showed, that 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. At this point we can only speculate that a weighted variance approach (that we sketched above) might work better than the plain variance approach. Moreover, it might be interesting to think about alternative ways of inferring the criticality from the environment.

VII. Related Work

Reinforcement learning based methods have recently shown great success in many domains, including Atari games [10], Go [13], and autonomous vehicles [11], [12]. 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 [9] utilized the human's intrinsic knowledge of the environment. An alternative reward shaping algorithm is the TAMER framework [8] and (the related Deep TAMER [18] 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 demonstrations. The Human-Agent Transfer algorithm [15] 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 [3].

Advice-based methods constitute another popular category of techniques 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 [5], [7].

At this point we turn our attention towards past work which is more closely related to our paper. 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 [16]. The importance of a state is defined by:

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

The paper 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 [1] 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 also want to 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 are to the bias-variance trade-off, which stems from the fact that the update of the Q-function suffers from a large bias if the value of n is small - and from large variance if it n is big. Various techniques have been developed to tackle this challenge.

De Asis [2] 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. [6] 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. [14] suggest an improvement of TD(λ) which achieves an effective bias reduction for the updates. This beneficial effect is a consequence of specific weights which 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 of doing this, it chooses the appropriate stepnumber 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, which speeds up the learning by controlling the variance of the updates.

VIII. Conclusions and Future Work

We presented the concept of criticality in reinforcement learning and sketched a number of ways how criticality could be defined. 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 the Atari Pong environment. The CVS agent, that was fed with a human-designed criticality function, was able to outperform such prominent competitors as DDQN and Monte Carlo. The CVS agent, that computed criticality from the environment via the plain variance, did not perform that well.

So far we presented an application of criticality in the domain reinforcement learning. Yet, the concept of criticality might also have applications in the context of human learning. Let us consider a learning scenario where the human student (for example a person who learns to play some game) is being assisted by an artificial intelligence agent (learning assistant). One of the ways the agent might support the student in his learning process, is by indicating to him, which situations are critical. When the student receives the information that a certain situation is critical, he pays more attention to it, and consequentially is more likely to master the challenging situation. The learning assistant might learn the criticality function from a human expert by calibrating a criticality model to fit a train set of samples provided by the expert. Beyond enabling the learning assistant to model the criticality function, the procedure of collecting a train set of critical situations might also boost 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 his vigilance on a high level throughout the complete duration of the task [4].

As we already mentioned above, the potential of the concept of criticality in reinforcement learning is not quite clear to us. The CVS algorithm merely links it to one tiny sub-domain of the reinforcement learning domain - that of n-step learning algorithms. We believe that criticality might bring substantial benefits to other areas of reinforcement learning as well. Moreover, it should be considered, that criticality - in the context of a more general definition, than the one we presented - might be utilized in other domains in artificial intelligence, even beyond the boundaries of the reinforcement learning domain.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3*, pages 617–624, 2011.

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

[17] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[18] Garrett Warnell, Nicholas R. Waytowich, Vernon Lawhern, and Peter Stone. Deep TAMER: interactive agent shaping in high-dimensional state spaces. *CoRR*, abs/1709.10163, 2017.