Article

Adaptive Behavior

Adaptive Behavior
1–12
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1059712319859650
journals.sagepub.com/home/adb
SAGE

Safe reinforcement learning using risk mapping by similarity

Jonathan Serrano-Cuevas[1], Eduardo F Morales[1] and Pablo Hernández-Leal[2]

Abstract

Reinforcement learning (RL) has been used to successfully solve sequential decision problem. However, considering risk at the same time as the learning process is an open research problem. In this work, we are interested in the type of risk that can lead to a catastrophic state. Related works that aim to deal with risk propose complex models. In contrast, we follow a simple, yet effective, idea: similar states might lead to similar risk. Using this idea, we propose risk mapping by similarity (RMS), an algorithm for discrete scenarios which infers the risk of newly discovered states by analyzing how similar they are to previously known risky states. In general terms, the RMS algorithm *transfers* the knowledge gathered by the agent regarding the risk to newly discovered states. We contribute with a new approach to consider risk based on similarity and with RMS, which is simple and generalizable as long as the premise *similar states yield similar risk* holds. RMS is not an RL algorithm, but a method to generate a risk-aware reward shaping signal that can be used with a RL algorithm to generate risk-aware policies.

Keywords

Reinforcement learning, risk, reward shaping, risk map, safe reinforcement learning

Handling Editor: Poramate Manoonpong, University of Southern Denmark, Denmark

1. Introduction

Reinforcement learning (RL) is an area of machine learning used to solve sequential decision making problems by means of interacting with the environment. This is achieved by allowing an agent to explore an unknown scenario in order to learn what is the best possible action to take given its current state. The goal of the learning process is to generate a policy, which is a mapping from states to actions that determines an optimal behavior, that is, return the maximum expected total reward (Otterlo & Wiering, 2012; Sutton & Barto, 1998). RL has been successfully applied in areas such as robotics (Kober, Bagnell, & Peters, 2013), Atari games (Mnih et al., 2013), and even traffic control (Pendrith, 2000). However, traditional RL is not effective for risk-critical applications (Geibel & Wysotzki, 2005), since it does not guarantee that the learned policy will not lead the agent to a cliff, crash the drone, cause the agent to run out of resources, or damage the agent in any way (Fulton & Platzer, 2018; Kahn, Villaflor, Pong, Abbeel, & Levine, 2017). This lack of consideration of risk constraints makes RL hard to use for solving tasks in business, finance, law, or any system where the behaviors can lead to large economic or human losses.

For the above-mentioned reasons, the concept of safe reinforcement learning (SRL; Leike et al., 2017) has recently gained interest. SRL can be defined as the process of learning policies that maximize the expected total return in problems in which it is important to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment processes (García & Fernández, 2015).

Current state of the art proposes solutions which make use of Gaussian processes to execute reachability analysis in continuous scenarios to learn risk-aware policies (Akametalu et al., 2014; Heidenreich, 2017). Nevertheless, we argue for a simpler, tabular approach

[1]Instituto Nacional de Astrofísica Óptica y Electrónica, México
[2]Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

Corresponding author:
Jonathan Serrano-Cuevas, Instituto Nacional de Astrofísica Óptica y Electrónica, Luis Enrique Erro #1, Tonantzintla C.P. 72840, Puebla, México.
Email: jonathan.serrano@inaoep.mx

that works if it can be assumed that similar states are similarly risky. With this idea in mind, we propose a risk mapping by similarity (RMS) algorithm that infers risk for new states based on the similarity to previously discovered states. The use of a similarity measurement allows the agent to anticipate how risky a new state is without actually experiencing a transition from this new state to a risky state. Our algorithm considers as well a parameter to tune how similar two states must be so that risk propagates. This parameter creates a safety region around known risky states. In general terms, the RMS algorithm *transfers* the knowledge gathered by the agent regarding the risk to newly discovered states, given the assumption that similar states are similarly risky. The benefits of transferring information to enhance the learning process has been shown vastly in previous literature (Munoz de Cote, Garcia, & Morales, 2016; Lazaric, Restelli, & Bonarini, 2008; Taylor, Jong, & Stone, 2008). The RMS output is a tunable reward shaping signal that can be used with a RL algorithm to learn risk-aware policies. The tuning allows control on (1) the type of similarity two states must hold so that risk propagates, (2) how this risk is propagated (linearly, exponentially, or any other), and (3) how different states must be so that they are not considered similar and prevent risk from propagating any further.

To prove the usefulness of RMS, we tested it in two different scenarios with two different similarity metrics and compared the risk maps generated with a recent state-of-the-art SRL algorithm. This work is divided as follows. First, the related work is described, then the RMS algorithm details are shown. Finally, the experimental results are shown with some final remarks and possible future work.

2. Related work

SRL is a requirement in many scenarios where the safety of the agent is particularly important, and, for this reason, researchers are paying increasing attention not only to the long-term reward maximization but also to damage and risk avoidance (Heger, 1994; Mihatsch & Neuneier, 2002).

SRL is a relatively new topic, and there are different definitions on what “safe” means. García and Fernández (2015) proposed a taxonomy which classifies the SRL techniques in two broad groups. The first group includes techniques which modify the optimality criterion. This implies modifying the reward signal (Grześ, 2017) to consider some risk measurement. Mahadevan and Maggioni (2007) and Morimura, Sugiyama, Kashima, Hachiya, and Tanaka (2012) propose related works that fit into the first category. Mahadevan constructs a graph connecting states that are nearby to create a diffusion model. This diffusion model is then used as a value function to learn a best policy given a set of basis functions. Our work aims to estimate a map of the inherent risk of a task instead of creating an alternative method to project value functions. Another difference is that the notion of similarity in RMS is flexible and can be set to any similarity metric that maps spaces in $\mathbb{R}^n$ to $\mathbb{R}$, that is, Manhattan distance (Black, 2006) or Euclidean distance. Morimura uses the concept of expected shortfall risk in order to estimate a *density* of rewards, which allows the algorithm to handle various risk-sensitive criteria.

Lipton et al. (2016) proposed a work that fits into the first category using a fear model, which estimates the probability that a given state would lead to a catastrophic outcome within fixed amount of steps k, using a neural network. This information is used to penalize dangerous states by means of a reward shaping process (Ng, Harada, & Russell, 1999). Whenever a catastrophe is reached at the nth step of an episode, the preceding k_r states are added to a danger buffer, while the first $n - k_r$ states of that episode are added to a safe buffer. Then, after each turn, in addition to updating a Deep Q-Network (DQN; Mnih et al., 2015), the fear model is updated by sampling 50% of the states from the danger buffer, assigning them a label of 1, and the remaining 50% from the safe buffer, assigning them a label of 0. This procedure deals with the *sisyphean curse* of DQN that causes the agent to forget about catastrophic states. Lipton’s approach and RMS both generate a risk model, the first using a neural network to estimate a danger zone and the latter using the concept of similarity.

The second group includes techniques which modify the exploration process through the incorporation of external knowledge or the guidance of a risk metric. Some related works that fit into this group are Akametalu et al. (2014) and Heidenreich (2017), which propose a method that makes use of the concept of reachability analysis. The method attempts to derive, using Gaussian processes and past observations, a disturbance function $d(x)$ to determine the limits of a region where exploration can be executed safely. A conservative disturbance is used to model the unknown dynamics of the system, and it is updated to increase the size of the safety set. Heidenreich makes use also of a safety-preserving control that returns the agent to a safe state. Instead of estimating a safe set, our approach aims to estimate the risky sets by inference, that is, estimate new risky regions by comparing newly found states to the already known risky states. Since we use a linear combination of reward and risk measurements to create risk-aware policies, our work fits into the first group. However, some similarities may be drawn with Akametalu et al. (2014), because they aim to expand the safe zone while we steer toward updating the risky zones.

3. RL

The goal of RL is to learn a policy π that selects, given its current state $s \in \mathcal{S}$, an action $a \in \mathcal{A}$ that will maximize its expected value $q_\pi(s,a)$ in the long run

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a\right]$$

where $\gamma \in [0,1]$ is a discount factor.

A policy π is said to be better than a policy π' if $q_\pi(s,a) \geqslant q_{\pi'}(s,a)$ for all $s \in \mathcal{S}$. π^* is said to be optimal if and only if $q_{\pi^*}(s,a)$ is better than or equal to all other policies

$$q^*(s,a) = max_\pi q_\pi(s,a)$$

A popular algorithm that keeps an updated estimate of q^* is Q-learning (Watkins & Dayan, 1992), which is the learning algorithm used in the experimental section.

4. Similarity measurements

Measuring similarity or distance between two data points is a core requirement for several data mining and knowledge discovery tasks that involve distance computation. For continuous data sets, the Minkowski distance is a general method used to compute distance between two multivariate points. In particular, the Minkowski distance of order 1 (Manhattan) and order 2 (Euclidean) are the two most widely used distance measures for continuous data (Boriah, Chandola, & Kumar, 2008). Since the purpose of this work is to show the usefulness of risk by similarity, we will use these two metrics in the experimental section. However, any other similarity measurement can be used with RMS.

5. Risk and SRL

The meaning of risk depends on the context it is used, nevertheless most of the risk definitions emanate from the financial or marketing (Peter & Ryan, 1976) literature. In SRL, the concept of risk as a fatal state is often used. This type of risk is measured as the probability for any given policy to reach an undesired state. A recent work that uses this type of risk is Lipton et al. (2016), which estimates risk using a classification algorithm that outputs the probability that a given state leads the agent to a fatal state over a fixed number of steps. The probability for each state creates a fear model F and is used to shape the reward signal used by a DQN network to learn a risk-aware policy. We also make use of risk as a fatal state concept; however, we aim to explore similarities between catastrophic or risky states to come up with the idea of risk by similarity, based on the premise that similar states exposes the agent to similar risk.

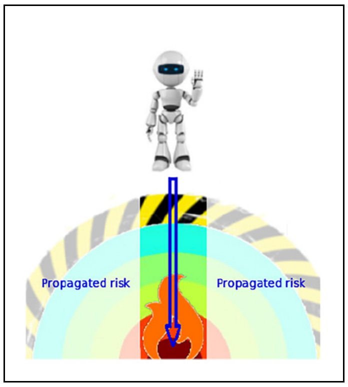


Figure 1. The idea behind risk mapping: when the agent finds a dangerous spot, the algorithm creates a safety region around it.

6. RMS

Traditional RL algorithms require an exploration process to determine which areas within the state space are undesired. If the agent visits state s and receives an undesired reward (i.e. low reward), this reward is propagated to the adjacent states as the exploration process finds more paths leading to s. When the agent chooses to exploit its knowledge, then these undesired states will be avoided. A disadvantage of this approach is that the agent must visit the undesirable states several times before learning how to avoid them, and this becomes an issue when the agent might be somehow damaged by visiting undesirable states. As an example, think about a robot which has to move from point A to point B. However, the route between these two points is not straightforward and has a fire located just in the middle; if the robot touches the fire, it wears out.[1] For the robot to learn where the fire is, it has to visit the fire several times. However, what if the agent could infer the presence of fire from the similarity of its current position to the position where it previously discovered there was fire? Figure 1 illustrates this idea. The agent discovered that there is fire at the bottom middle by following the trajectory described by the blue arrow. So far, the agent only knows the unshaded area in the middle; however, it now knows that any point at the same distance of the fire is similarly risky. Then, as the agent discovers the area in yellow and black, the fire risk will propagate and the agent will avoid the area, creating a risky region. This is the idea behind RMS: similar states yield similar effects. So, the main issue is to use a proper measurement that reflects how similar two states are and then define risk as a function of the similarity.

By doing this, the risk learned in one episode is transferred to subsequent episodes to discourage the agent to visit previously discovered risky states. In a

Algorithm 1. RMS Algorithm.

```
1: procedure INITIALIZE
2:   # Hyper parameters
3:   $Disc(s, r, s')$                                   ▷ A discriminator function that labels as risky or not a transition
4:   $I_n \leftarrow c_1$                                ▷ Influence threshold
5:   $Sim(s, s')$                                       ▷ A similarity measurement between two states
6:   $f(Sim)$                                           ▷ A risk function with an output between 0 and 1
7:   $R_D \leftarrow 0$                                  ▷ The default total risk value
8:   # Initialize algorithm variables
9:   $\hat{k} \leftarrow \emptyset$                      ▷ The risky states vector
10:  $V \leftarrow \emptyset$                            ▷ Vector of known states
11:  $M_{\hat{k}} \leftarrow \hat{k} \times V$            ▷ A similarity matrix
12:  $N \leftarrow c_2$                                  ▷ The number of episodes
13:  $t \leftarrow 0$                                    ▷ The initial time
14:  $s \leftarrow$ initial state, $a \leftarrow$ initial action
15:  while $t < N$ do:
16:      Execute action $a$ while in state $s$ according to a policy
17:      Record the reward $r$ and the next state $s'$
18:      if $s \notin V$ then                            ▷ If state s was not visited before
19:          Add $s$ to $V$                              ▷ Add to the vector of known states
20:      if $Discriminator((s, r, s'))$ == true AND $s' \notin \hat{k}$ then   ▷ If state is risky and is not in $\hat{k}$
21:          Add $s'$ to $\hat{k}$                        ▷ Add state to the risky states vector
22:          End Episode                                 ▷ Add to the list of risky states
23:      # Update the risk propagating from risky states to known states.
24:      for each $k \in \hat{k}$ do
25:          for each $j \in V$ do
26:              if $Sim(k, j) < I_n$ then               ▷ If j is similar enough to k
27:                  $M_{\hat{k}}[k][j] \leftarrow f(Sim(k, j))$   ▷ Propagate risk
28:              else
29:                  $M_{\hat{k}}[k][j] \leftarrow R_D$    ▷ No risk is propagated
30:  # $M_{\hat{k}}$ is a matrix that contains the risk propagating from risky states to all known states.
31:  $R(s = j) \leftarrow \sum_{i=0}^{|k|} M_{\hat{k}}[i][j]$   ▷ The sum of j column corresponds to the risk propagated to state $s = j$
32:  # RMS algorithm ends here. Now R is used to update Q-values using $r'$ instead of r.
33:  $r'(s, a, s') = r(s, a, s') - \beta R(s')$
34:  $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
35:  $s \leftarrow s'$
36:  Choose $a$ according to $\epsilon$-greedy.
```

nutshell, we *transfer* the risk knowledge gathered by the agent to newly discovered states, assuming that similar states are similarly risky as well. The RMS output is a risk map, which is a per-state reward shaping function proportional to each state's risk. So, RMS is not an RL algorithm, but a tool that can be used with an RL algorithm to learn risk-aware policies. Some of the features of the RMS algorithm are as follows:

- Since RMS builds the risk map online while exploring the environment, it does not require any prior knowledge about the state transitions.
- RMS works for n-dimensional states and is agnostic to any similarity measurement.
- The similarity between two states does not change with time, hence RMS is time independent.

6.1. RMS parameters

Algorithm 1 describes the RMS, and its parameters are as follows:

- $Disc(s, r, s')$ is a boolean function that determines if a given transition must be considered risky. As an example, *Disc* will evaluate to *True* in a grid world if $s' \in \{DangerStates\}$; however, a criteria such as shortfall risk can be used to determine if a state is risky.
- $\hat{k}$ is the risky states list.
- V is the list of visited states.
- r_{thres} is the reward threshold to consider a state risky. If a transition to a state s' provides a reward smaller than this value, it will be added to the risky states list $\hat{k}$.
- R_D is the default risk for any state whose risk information is unknown.
- $Sim(s, s')$ is a similarity measurement between two states such as Manhattan distance, Minkowski distance, or, its particular case, the Euclidean distance.
- $f(Sim(s, s'))$ is a risk function that takes the similarity measurement as an input and determines how the risk is propagated toward similar states. This allows the risk to be decreased linearly, exponentially, or in any way required by the problem.

- I_n is an influence threshold that limits the effect of $f(Sim(s, s'))$. Beyond this limit, $f(Sim(s, s'))$ is zero. In order for a pair of states (k,j) to exist in $M_{\hat{k}}$, its similarity measure has to be less than this value. An important feature of RMS is that if any given state s_a is within the influence area I_n of another two dangerous states s_b and s_c, the risk of both states will propagate toward s_a.
- $M_{\hat{k}}$ is a similarity sparse matrix where the columns (indexed by j) are each one of the visited states and the rows (indexed by k) are the set $\{x|x = Sim(k,j)<I_n\}$.
- β is a user-defined multiplier that sets the size of the risk penalization.
- c_1 and c_2 are hyper parameters.

The core of RMS are the $Sim(s, s')$ and $f(Sim(s, s'))$ parameter functions. The former quantifies the similarity between two given states s and s', while the latter transforms similarity into a risk measurement. Both functions are to be chosen according to the problem. If *Sim* is chosen as Euclidean distance, then, for any given pair of states, the similarity metric will increase as these become more different or distant. Hence, $f(Sim)$ must be a monotonic decreasing function so that for two distance measurements $Sim(s, s') = a$ and $Sim(s, s'') = b$ where $a<b$, $f(Sim(s, s'')) < f(Sim(s, s'))$ holds. If $Sim(s, s') \geqslant I_n$, then the risk $f(Sim(s, s')) = 0$. Figure 2 shows an example of these relationships. The red area represents the risk propagating from a state to other similar states. Please refer to Appendix 1 for a more in-depth explanation of $f(Sim)$ and an additional example.

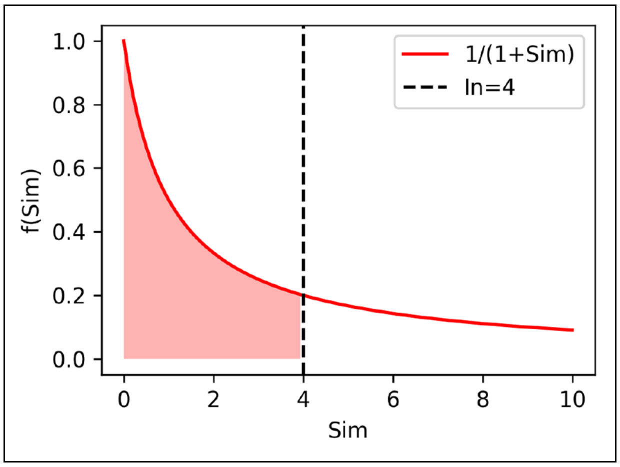


Figure 2. An example of the relationship between $Sim(s, s')$, $f(Sim(s, s'))$, and I_n.
If the similarity measurement is less than I_n, then the risk will be defined by $f(Sim)$. In this example, $f(Sim) = 1/1 + Sim$, and $I_n = 4$ limits how far the risk propagates.

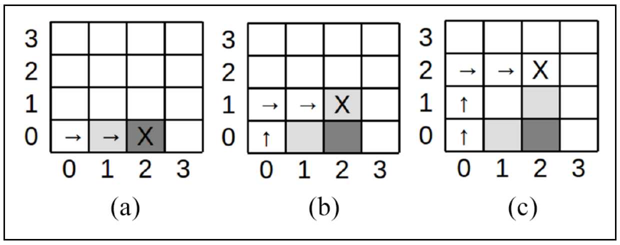


Figure 3. An example of the risk mapping process across three episodes ((a) to (c)) with an influence threshold $I_n = 1$ and Manhattan distance as a measurement of distance. The dangerous states are dark shaded, and the propagated risk is light shaded. In episode (b), the risk propagated from cell (0,2) to cell (1,2) even though there was no transition from the latter to the former. In episode (c), the agent followed a path that took it further than the I_n threshold, and then nothing new happened.

6.2. RMS description

Lines 2 through 14 initialize the variables, and in line 15, the risk mapping loop starts. In line 16, an action is chosen and the next line records the new state and the obtained reward. Line 20 evaluates if the new state s' is risky by evaluating $Disc(s, r, s')$ and if it is not already in $\hat{k}$. If these conditions are met, then the new state s' is added to $\hat{k}$. Lines 24 through 29 update $M_{\hat{k}}$, discarding any pair which does not comply with the I_n restriction. Both $f(Sim)$ and I_n control how the risk propagates, that is, by modifying $f(Sim)$, the risk could propagate following a decreasing exponential pattern, as long as $Sim<I_n$. This feature also reduces the number of elements stored in $M_{\hat{k}}$. Then, line 31 determines the current risk estimate of the state s. Line 33 executes the reward shaping process, and finally, line 34 execute the Q-learning update rule. The only difference is the use of r' instead of r. The main advantage of *RMS* is its flexibility, because any other reward shaping function can be considered, by modifying line 33, and any other learning algorithm can be used, by modifying line 34. It is important to recall that, as shown in line 33, RMS used with an RL algorithm such as Q-learning will modify the actual reward for certain states considered risky. This will effectively modify the learned policy as it will be shown in the experimental section, which is what we intend. The RMS output is upper-bounded by I_n and β and is deterministic, hence it will not affect the convergence properties of any RL algorithm it is used with.

6.3. RMS example

Figure 3 shows an brief example of RMS algorithm across three episodes in a 4 by 4 grid world. In episode (a), the agent finds the dangerous state (0,2). So far, it knows as well states (0,0) and (0,1), hence $V_a = \{(0,0),(0,1),(0,2)\}$ and $\hat{k}_a = \{(0,2)\}$. Since only state (0,1) is at distance 1 of the known risky state, it receives propagated risk, hence it is now light shaded.

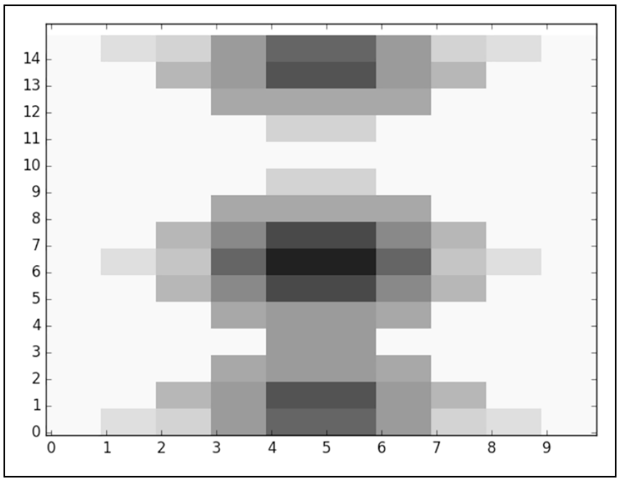


Figure 4. An example of the risk map generated by RMS in a grid world with three risky areas in rows 0, 6, and 14.

In the next episode (b), the agent finds new states across row 1. Now, $V_b = V_a \bigcup \{(1,0),(1,1),(1,2)\}$, and since no new risky state was discovered, $\hat{k}_b = \hat{k}_a$. During episode (b), state (1,2) is discovered to be at a distance of 1 from the known risky state. For this reason, the risk is propagated toward state (1,2). The agent knows this state is dangerous without transitioning from state (1,2) to state (0,2) because of the propagated risk. This knowledge is used in episode (3), where the agent keeps away from the risky states by moving across row 2. Figure 4 shows another example of a RMS output in a 9×14 grid world with dangerous states in rows 0, 6, and 14. After a random exploration of 100 steps, RMS manages to create a risk map, where the darkest cells are the most risky.

7. Experiments

This section shows the results of using RMS to infer risk. A simple grid world is used to explain the impact of RMS.

7.1. *Cliff world scenario*

The grid world we used is shown in Figure 5. This is a 4×10 variation of Sutton & Barto's (1998) cliff world. The agent starts in cell *A*, and its task is to find cell *G* while avoiding the cliff cells. The agent can move up, right, down, and left, with a 10% probability of a random action. The cliff cells and the *G* cell are final states. The cliff cells provide the agent with a reward of -10 and the *G* cell with a reward of 10. Any other cell yields a reward of -1.

Since we are concerned with risk, we would like RMS to build a safety area around the risky states in row 0 as soon as possible. Before allowing RMS to estimate the risky zone, some decisions have to be made,

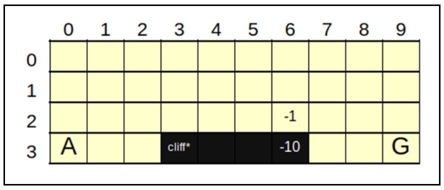


Figure 5. Modified cliff world.
Start cell is [3,0], cliff is in row 3, and Goal cell is [3, 9].

Table 1. RMS settings for cliff world.

Parameter	Value
Sim	Manhattan distance
$f(Sim)$	$1/(1 + f(Sim))$
I_n	2
β	-10
Disc	True if $r > -1$ else False

Only a value $I_n = 2$ was selected because of the scenario size. If $I_n > 2$ was chosen, there would be no safe path for the agent to move across row 0.

namely, a similarity measurement *Sim*, a risk measurement function $f(Sim)$, a discriminator function *Disc*, and the influence threshold I_n. The selection for these parameters are shown in Table 1.

From Table 1, two of the most important features of RMS can be derived: the similarity function is independent of the risk measurement, and the *power* of the risk propagation can be controlled using I_n, which is expressed in the same units as *Sim*, and is time independent. The risk propagation depends only on these two parameters. Figure 6(a) shows the output of RMS algorithm after 30 episodes. The dangerous cliff cells are surrounded by a security area that is used as a shaping function for the Q-learning reward signal. As a direct comparison, Figure 6(b) shows the output of the fear model proposed by Lipton et al. (2016) after 1500 episodes. The shape of the security area is similar to the RMS output with some differences. The most remarkable is the lack of symmetry of the security area, which means that the fear model estimates that two equally separated states from the cliff are not necessarily similarly risky and the existence of a very dangerous cell in (3,2), just next to the cliff. Both outcomes are related to the fact that the fear model is generated by pulling random samples of the safe and dangerous state sets, and these sets are generated as the agent follows different paths, or trajectories. Hence, the fear model depends on the trajectories available during the training. Since the agent has to move from left to right to reach the cell goal, there is a large probability that cell (3,2) leads to the cliff compared to cell (3,7). Since RMS estimates risk based on a fixed similarity measurement, these biases do not occur.

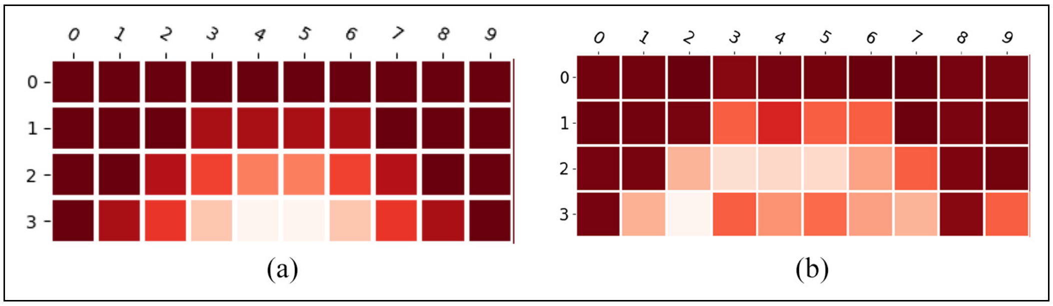


Figure 6. A direct comparison between Lipton's state-of-the-art approach to deal with risk and RMS: (a) Risk map generated by RMS after 30 episodes in cliff world with $I_n = 2$ using Manhattan distance as a similarity measurement. The darkest cells are the safer. In this particular example, a value $I_n < 3$ has to be used, otherwise there would be no safe path across row 0 and (b) Fear model generated after 1500 episodes as described by (Lipton et al., 2016). Both RMS and Lipton's fear model estimate the safe path to be across row 0.

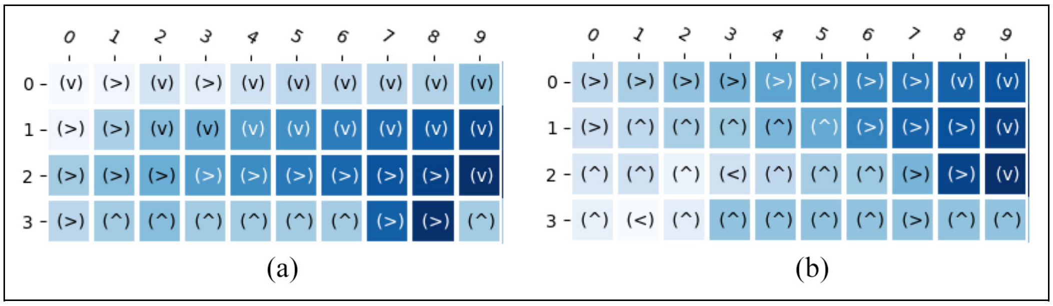


Figure 7. Policy heatmap learned for the beach scenario without RMS (a) and with RMS (b). The darker cells are the most desirable: (a) If no RMS is used, the learned policy moves the agent next to the cliff and (b) When using RMS, the most desirable path is across row 0, moving away from the cliff as possible.

Following with the RMS effects, Figure 7(a) shows the policy followed by an agent which is not preventing risk in any way. As expected, the agent runs across row 1, just next to the cliff. On the contrary, Figure 7(b) shows the policy learned by an agent using RMS. The agent moves far away from the cliff and crosses the scenario from left to right through row 0. A similar policy plot is obtained with the fear model.

Figure 8 shows the effect of using RMS to control risk in terms of failure rate, compared to plain Q-learning (*no RMS*), *SARSA*, and Lipton's fear model (*DQN* + *Lipton*). Compared to *no RMS* and *SARSA*, RMS manages to reduce the failure rate to almost zero. This occurs after 170 episodes. A comparison is made as well between RMS with $I_n = 2$ and $I_n = 4$. In terms of failure count, $I_n = 2$ has a lower failure rate. This is because with $I_n = 4$, RMS estimates that every single cell, except for the ones at the upper corners, is risky. Hence, the agent has no big incentive to move away from the cliff, causing it to fail more and receive a lower overall reward compared to $I_n = 2$. On the contrary, *DQN* + *Lipton* also learns a policy similar to the one shown in Figure 7(b); however, this happens after 1500 episodes due to the fact that the latter algorithm makes use of two neural networks to estimate the fear model and the policy.

As mentioned earlier, the core idea of RMS is to estimate a risk map based on similarity. Lipton's fear model does the same job by estimating the probability of failure given a threshold of steps in the future, that is, answering the question *Provided that I am in this state, whats the probability of reaching a catastrophic state in n steps?* Hence, a direct comparison between *RMS* and *DQN* + *Lipton* can be made by observing Figure 6, which shows the risk map generated by the two algorithms in the grid world. Both algorithms estimate that there is a safe zone across row 0 using different approaches. RMS learns this by observing that the states in row 0 are the most different compared to the dangerous states in row 3. Lipton's fear model infers that the least probability of visiting the dangerous states occurs while in row 0. The result is the same: while including this knowledge using an RL algorithm, the agent is forced to move across row 0, far away from the danger zone.

Even though we do not aim to improve the learning speed or the final reward of any RL algorithm, for the sake of completeness, Figure 9 shows the reward plot

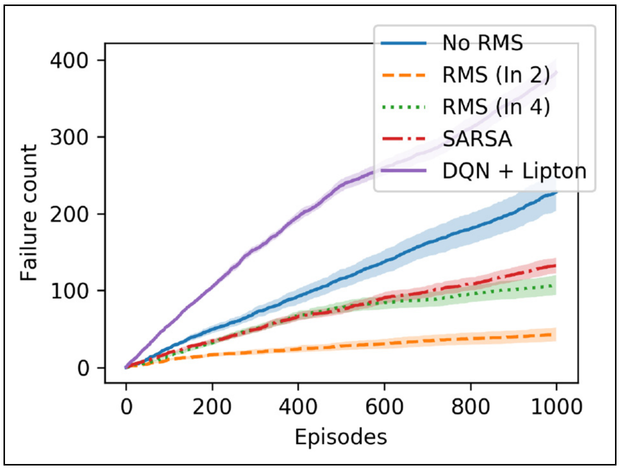


Figure 8. Accumulated failure for no RMS, RMS (I_n = 2), SARSA, and DQN + Lipton in cliff scenario.

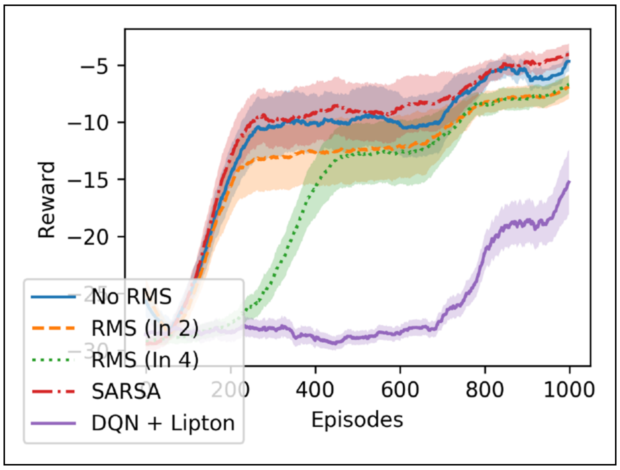


Figure 9. Rewards plot for no RMS, RMS (I_n = 2), SARSA, and DQN + Lipton in cliff scenario.

for the experiments shown in Figure 8. The *RMS* plot obtains a shorter episodic reward because it follows a longer path across row 0; however, it is more stable since it does not fall off the cliff. *DQN + Lipton* reaches a final episodic reward of -7.2, similar to *RMS*, but after 1500 episodes. Table 2 shows the paremeters used for Q-learning and SARSA experiments.

Table 2. Q-learning/SARSA settings.

Parameter	Value
α	0.9
γ	0.1
ϵ	Decreasing from 1.0 to 0

7.2. *Cartpole scenario*

To test RMS using a different similarity measurement, AI Gym's (Brockman et al., 2016) cartpole scenario was used. The cartpole scenario consists of a cart that moves along the horizontal axis and a pole that is anchored on the cart. At every time step, the position x, velocity x', angle θ, and angular velocity θ' can be observed. In any given state, the cart has only two possible actions: move to the left or move to the right. Thus, the cartpole has four dimensions of continuous values, and the action-space has one dimension of two discrete values. In the cartpole scenario, an episode starts with the pole in a vertical position and the cart at the middle and ends if $|x| > 2.4$ or $|\theta| > 12^\circ$. For every step, if the cartpole is within these bounds, a reward of $+1$ is received, and the scenario is considered completed when the average reward of the last 100 episodes is larger than 195. From this, it can be said that it is important to prevent the pole to get anywhere near the danger zone of $\pm 12^\circ$, and this can be achieved using RMS. Such danger zones exist in cells (1, 0) and (4, 11). To accomplish this, the continuous variables θ and θ' were discretized into a series of buckets. Then, RMS was used with the settings shown in Table 3.

The *Disc* function was set directly equal to the end condition of the cartpole scenario. So, any state similar in terms of Euclidean distance will be labeled as dangerous. To show the RMS results, the buckets were mapped into a grid, which is shown in Figure 10. For both subplots, the buckets of θ are in the rows and the buckets of θ' are in the columns. We expect that RMS is able propagate the risk from dangerous states to *inner* states.

Table 3. RMS settings for cartpole.

Parameter	Value
Sim	Euclidean distance
f(*Sim*)	$1/(1 + f(Sim))$
I_n	2, 4
Disc	True if $\lvert\theta\rvert > 12^\circ$ else False
β	-1

Figure 10(a) shows the risk map generated when $I_n = 2$ and Figure 10(b) shows it for $I_n = 4$. These plots show one of the benefits of being able to tune how aggressive the risk map is. For small values of I_n, RMS only penalizes very similar states to the risky ones, and for larger values, states less similar are penalized as well. For this reason, the cells further from the middle columns are whiter in Figure 10(a) compared to Figure 10(b). Figure 10(c) shows the risk map generated in cartpole scenario using Lipton's fear model. All three subfigures shown in Figure 10 estimate correctly that the danger states (when $|\theta| > 12^\circ$) occur in the first column of row 1, which we will call D_1, and the last column of row 4, or D_2. Both models estimate that the cells to the right of D_1 and to the left of D_2 are

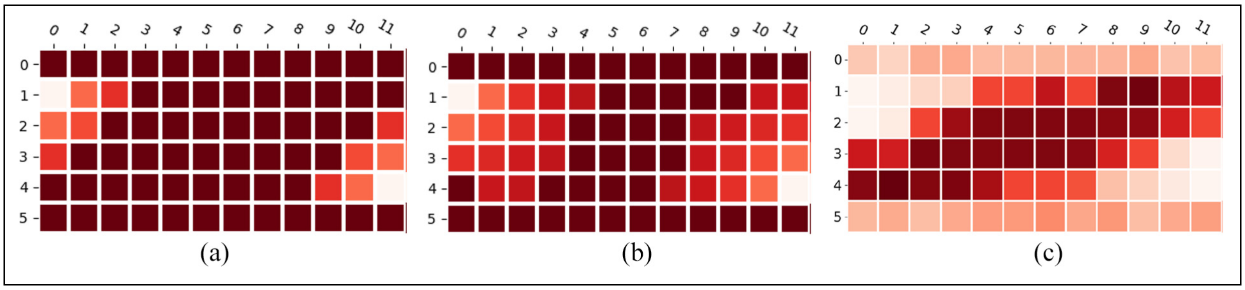


Figure 10. Risk maps of a discrete version of cartpole scenario state space using both RMS with different settings and Lipton's fear model. For the three experiments, the darker cells are the safer: (a) With I_n = 2, the risk map has a shallower influence, (b) With I_n = 4, the risk map has a deeper influence, and (c) Risk map generated by Lipton's fear model.

dangerous. This is because it is worst if the pole is close to $\pm 12^{\circ}$ moving fast, than moving slow.

For RMS, the effect of using different values of I_n is visible in Figure 11, which shows the failure count in 1000 episodes for different I_n values: when using $I_n = 4$, the failure count drops from 130 on average to 90 occurrences, with an average failure rate of less than 10 per 1000 episodes. Lipton's fear model achieves a similar failure rate at 3000 episodes, due to the fact that the model's neural networks require more samples to train. This is just a models' feature and not a disadvantage.

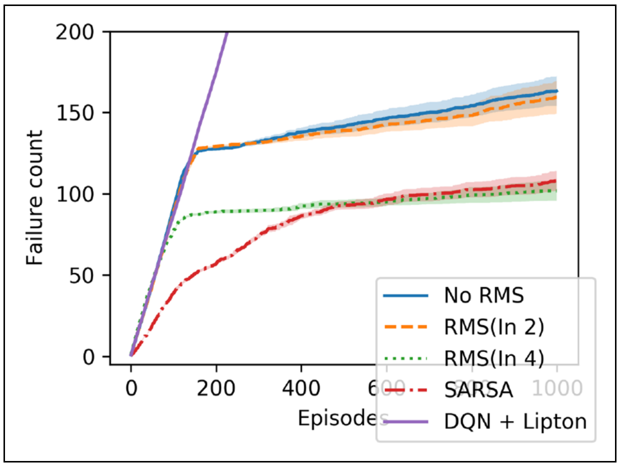


Figure 11. Accumulated failure plot for no RMS, RMS, and Fear in cartpole scenario.

7.3. Discussion

Our experiments showed that propagating the risk from a risky state to newly discovered states, proportionally to a similarity metric, helps the agent to create safety regions of adjustable size around undesired states. The premise for this to work is that similar states yield a similar risk. This premise holds in many scenarios: a robot navigating a terrain may detect a change in slope before finding a ditch, a trading market may send signals before a crash, a navigation signal may detect a car is getting closer to it before an accident occurs, or an agent leaving a bomb in Pommerman (Resnick et al., 2018). In any case, choosing an appropriate similarity measurement and risk function is paramount, and RMS allows both parameters to be adjustable.

8. Conclusion

In this work, we explored a simple idea: risk can be inferred from similarity to previous known states. With this in mind, we presented RMS, a tabular algorithm to account for risk by propagating implicitly the risk from previously known dangerous states in a proportional way to a similarity metric. If the premise *similar states yield similar risk* holds, then RMS will be able to learn a risk map online that can be used to develop safe policies. We tested our algorithm in a grid world and in a continuous scenario proposed by SRL literature, using two different similarity metrics. We compared the risk map generated by RMS to a recent state-of-the-art SRL algorithm and pointed out some advantages of RMS. We experimented with two similarity metrics, but showed that both the similarity measurement *Sim* and the risk function $f(Sim)$ are fully customizable in RMS, as long as it can be represented with a scalar value. This feature makes the idea of risk by similarity extendable to other more general environments. To accomplish this, a suitable similarity metric has to be defined, pretty much in the same way a loss function has to be defined in an optimization problem.

Some future work is to modify RMS so that I_n parameter is learned in terms of how the risk differs for two similar states, that is, if there is a ditch with no slope, I_n might be tuned to a large number so an artificial slope is created. Another future path is propagating the risk not in terms of the shortest distance from s to s', but in terms of the shortest path between two states. This becomes obvious if we observe that even if two states s and s' are close in terms of their features, hence are similar, there may be no single action, or no possible series of actions, that connects them; that is, $P(s, a, s') = 0$. Nevertheless, if s is considered as risky, RMS will propagate this risk toward s'.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was conducted with support of a research scholarship from Consejo Nacional de Ciencia y Tecnología (CONACYT).

ORCID iD

Jonathan Serrano-Cuevas https://orcid.org/0000-0003-4232-4209

Note

1. We deliberately picked the word wear out instead of damage to allow the idea that even though fire is undesirable, it does not instantly *kill* the agent. To completely avoid risky states, some sort of external knowledge is required (Abbeel et al., 2009; Driessens & Džeroski, 2004).

References

Abbeel, P., Coates, A., Hunter, T., & Ng, A. (2009). Autonomous autorotation of an RC helicopter. In O. Khatib, V. Kumar, & G. J. Pappas (Eds.), *Experimental robotics* (pp. 385–394). Berlin, Germany: Springer.

Akametalu, A. K., Kaynama, S., Fisac, J. F., Zeilinger, M. N., Gillula, J. H., & Tomlin, C. J. (2014). Reachability-based safe learning with Gaussian processes. In *2014 IEEE 53rd annual conference on decision and control* (CDC) (pp. 1424–1431). New York, NY: IEEE.

Black, P. E. (2006). Manhattan distance. *Dictionary of Algorithms and Data Structures*, *18*, Article 2012.

Boriah, S., Chandola, V., & Kumar, V. (2008). Similarity measures for categorical data: A comparative evaluation. In C. Apte, H. Park, K. Wang, & M. J. Zaki (Eds.), *Proceedings of the 2008 SIAM international conference on data mining* (pp. 243–254). Philadelphia, PA: SIAM.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI gym. Retrieved from https://arxiv.org/abs/1606.01540

Driessens, K., & Džeroski, S. (2004). Integrating guidance into relational reinforcement learning. *Machine Learning*, *57*, 271–304.

Fulton, N., & Platzer, A. (2018, February2–7). Safe reinforcement learning via formal methods. *Paper presented at Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA.

García, J., & Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, *16*, 1437–1480.

Geibel, P., & Wysotzki, F. (2005). Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, *24*, 81–108.

Grześ, M. (2017). Reward shaping in episodic reinforcement learning. In K. Larson, M. Winikoff, S. Das, & E. Durfee (Eds.), *Proceedings of the 16th conference on autonomous agents and multiagent systems* (pp. 565–573). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Heger, M. (1994). Consideration of risk in reinforcement learning. In W. W. Cohen, & H. Hirsh (Eds.), *Proceedings of the eleventh international conference on machine learning* (pp. 105–111). San Francisco, CA: Morgan Kaufmann.

Heidenreich, C. (2017). *Safe learning for control: Combining disturbance estimation, reachability analysis and reinforcement learning with systematic exploratio*n (Master's thesis). Retrieved from https://kth.diva-portal.org/smash/get/diva2:1140173/FULLTEXT01.pdf

Kahn, G., Villaflor, A., Pong, V., Abbeel, P., & Levine, S. (2017). *Uncertainty-aware reinforcement learning for collision avoidance* (arXiv: 1702.01182). Retrieved from https://arxiv.org/abs/1702.01182

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

Lazaric, A., Restelli, M., & Bonarini, A. (2008). Transfer of samples in batch reinforcement learning. In W. Cohen, A. McCallu, & S. Roweis (Eds.), *Proceedings of the 25th international conference on machine learning* (pp. 544–551). New York, NY: ACM.

Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., & Legg, S. (2017). *AI safety gridworlds* (arXiv: 1711.09883). Retrieved from https://arxiv.org/abs/1711.09883

Lipton, Z. C., Azizzadenesheli, K., Kumar, A., Li, L., Gao, J., & Deng, L. (2016). Combating reinforcement learning's Sisyphean curse with intrinsic fear (arXiv: 1611.01211). Retrieved from https://arxiv.org/abs/1611.01211

Mahadevan, S., & Maggioni, M. (2007). Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, *8*, 2169–2231.

Mihatsch, O., & Neuneier, R. (2002). Risk-sensitive reinforcement learning. *Machine Learning*, *49*, 267–290.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning (arXiv: 1312.5602). Retrieved from https://arxiv.org/abs/1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., & … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., & Tanaka, T. (2012). Parametric return density estimation for reinforcement learning (arXiv: 1203.3497). Retrieved from https://arxiv.org/abs/1203.3497

Munoz de Cote, E., Garcia, E. O., & Morales, E. F. (2016). Transfer learning by prototype generation in continuous spaces. *Adaptive Behavior*, *24*, 464–478.

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In I. Bratko, & S. Dzeroski (Eds.), *Proceedings of the sixteenth international conference on machine learning (ICML)* (pp. 278–287). San Francisco, CA: Morgan Kaufmann.

Otterlo, M., & Wiering, M. (2012). Reinforcement learning and markov decision processes. *Reinforcement Learning*, *12*, 3–42.

Pendrith, M. D. (2000). Distributed reinforcement learning for a traffic engineering application. In C. Sierra, M. Gini, & J. S. Rosenschein (Eds.), *Proceedings of the fourth international conference on autonomous agents* (pp. 404–411). New York, NY: ACM.

Peter, J. P., & Ryan, M. J. (1976). An investigation of perceived risk at the brand level. *Journal of Marketing Research*, *13*, 184–188.

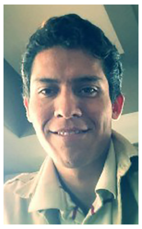
Resnick, C., Eldridge, W., Ha, D., Britz, D., Foerster, J., Togelius, J., & Bruna, J. (2018). Pommerman: A multiagent playground (arXiv: 1809.07124). Retrieved from https://arxiv.org/abs/1809.07124

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (*Vol. 1*). Cambridge, MA: The MIT press.

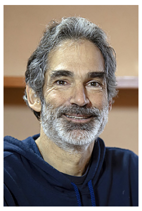
Taylor, M. E., Jong, N. K., & Stone, P. (2008). Transferring instances for model-based reinforcement learning. In W. Daelemans, B. Goethals, & K. Morik (Eds), *Joint European conference on machine learning and knowledge discovery in databases* (pp. 488–505). Berlin, Germany: Springer.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

About the Authors



Jonathan Serrano-Cuevas is assistant researcher at INAOE, Mexico. He is interested in applied reinforcement and deep learning.



Eduardo F Morales received his PhD from the Turing Institute at Strathclyde University in Glasgow, UK. He has participated as researcher at the Electric Power Research Institute in California, and New Wales University in Australia. Currently he is a researcher at INAOE, México. His research interests are on machine learning, relational reinforcement learning and robotics.

Pablo Hernandez-Leal is a researcher at CWI, the national research institute for Mathematics and Computer Science of the Netherlands. Pablo is interested in multiagent systems, reinforcement learning and game theory.

Appendix 1

Here, additional results are shown when smooth, average, and steep $f(sim)$ functions are used. The smooth function is $f(Sim) = -0.1Sim + 1$, the average function is $f(Sim) = 1/(1 + Sim)$, which corresponds to the one used in all the cliff world and cartpole scenarios, and the steep function is $f(Sim) = 1/(1 + 10Sim)$. Figure 12 shows the source $f(Sim)$ functions, Figure 13 shows the generated risk maps, and Figure 14 shows the resulting policies. Now, we will focus on explaining the differences between the risk map figures.

Both were generated with $I_n = 2$, which means that the risk will propagate only to cells two steps away from the cliff. As in the previous risk maps, the darker the cell the safer it is. In Figure 12(c), the risk decreases fast as Sim increases, so fast that even before $Sim = I_n$, the risk is almost zero. The results of this can be seen in Figure 13(b), where the risky states, as described by the risk map, are only the cliff cells and the surrounding cells, which are slightly clearer than the zero-risk cells.

To clarify even further the effects of using one or another $f(Sim)$ functions, Figure 14 shows the resulting policies. For both the smooth and average $f(Sim)$

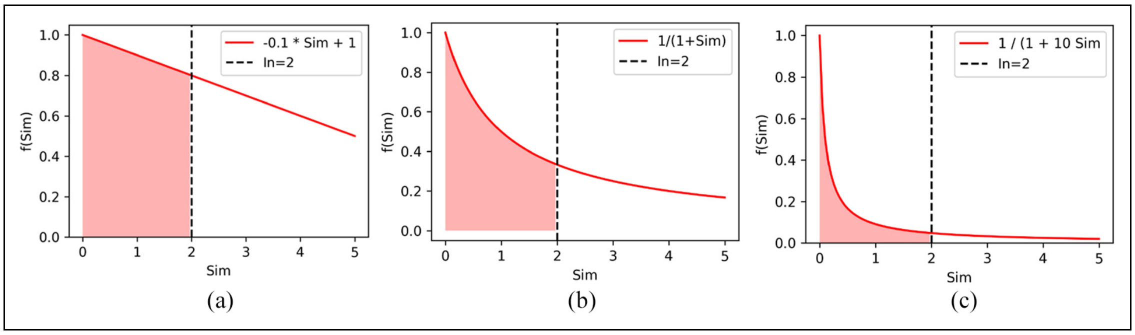


Figure 12. An effect comparison of using different *f(Sim)* functions with $I_n = 2$: (a) A smooth *f(Sim)* function: it decreases slowly as *Sim* increases, (b) An average *f(Sim)* function, and (c) A steep *f(Sim)* function: it decreases fast as *Sim* increases.

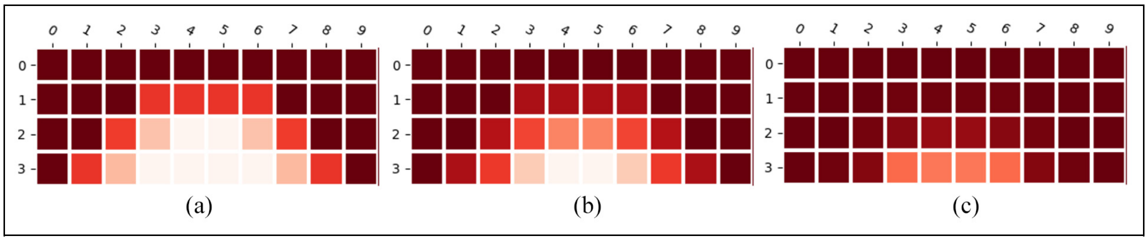


Figure 13. The corresponding risk maps: (a) the smooth function risk map, (b) the average function risk map, and (c) the steep function risk map.

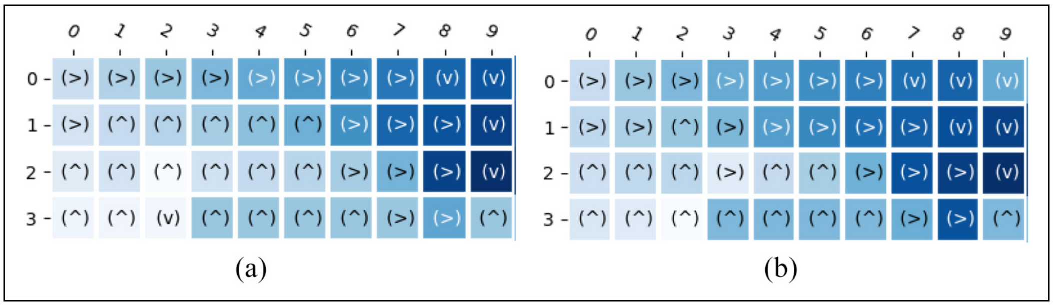

0 - (>) (>) (>) (>) (>) (>) (>) (>) (v) (v)
1 - (>) (^) (^) (^) (^) (^) (>) (>) (>) (v)
2 - (^) (^) (^) (^) (^) (^) (>) (>) (>) (v)
3 - (^) (^) (v) (^) (^) (^) (^) (>) (>) (^)
(a)
0 - (>) (>) (>) (>) (>) (>) (>) (v) (v) (v)
1 - (>) (>) (^) (>) (>) (>) (>) (>) (v) (v)
2 - (^) (^) (^) (>) (^) (^) (>) (>) (>) (v)
3 - (^) (^) (^) (^) (^) (^) (^) (>) (>) (^)
(b)

Figure 14. The generated policies. The arrows in each cell show the action with the largest Q-value, accounting for risk: (a) The smooth and average *f(Sim)* function policy and (b) The steep *f(Sim)* function policy.

functions, the corresponding policy is shown in Figure 14(a). In this figure, the risk propagating from the cliff cells forces the agent to move across row 0, as far away as possible from the danger. This path can be seen as the zero-risk darker cells in Figure 13(a) and (b). On the contrary, when the steep $f(Sim)$ function is used, the propagating risk decreases fast and allows the agent to move closer to the danger zone. Figure 14(b) reflects this fact.

These additional figures expect to show that the $f(Sim)$ functions propagate risk concentrically from the danger to the safe zones only if $Sim \leq I_n$. How *fast* this occurs depends on $f(Sim)$ and how far away depends on I_n.