Engineering Applications of Artificial Intelligence 88 (2020) 103360

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Teaching a humanoid robot to walk faster through Safe Reinforcement Learning☆

第一次将safe RL应用到实际的机器人当中

Javier García *, Diogo Shafie

DEI/FEUP – Informatics Engineering Department, Faculty of Engineering of the University of Porto, Portugal

ARTICLE INFO

Keywords:
Safe Reinforcement Learning
Biped walking

将safe RL应用到人形机器人控制上

已经有一个策略可以让机器人安全地行走

ABSTRACT

Teaching a humanoid robot to walk is an open and challenging problem. Classical walking behaviors usually require the tuning of many control parameters (e.g., step size, speed). To find an initial or basic configuration of such parameters could not be so hard, but optimizing them for some goal (for instance, to walk faster) is not easy because, when defined incorrectly, may produce the fall of the humanoid, and the consequent damages. In this paper we propose the use of Safe Reinforcement Learning for improving the walking behavior of a humanoid that permits the robot to walk faster than with a pre-defined configuration. Safe Reinforcement Learning assumes the existence of a safe baseline policy that permits the humanoid to walk, and probabilistically reuse such a policy to learn a better one, which is represented following a case based approach. The proposed algorithm has been evaluated in a real humanoid robot proving that it drastically increases the learning speed while reduces the number of falls during learning when compared with state-of-the-art algorithms.

1. Introduction

For a humanoid robot, performing tasks in complex environments requires fast and stable behaviors. Humanoid walking is one of the most interesting research topics and an important application area for multi-disciplinary fields. For instance, machine learning has provided many improvements in that task (Meriçli and Veloso, 2010; Farchy et al., 2013; Gil et al., 2019). However, in all these works, the risk of a robot fall is not explicitly tackled, and no explicit definition of risk is ever given nor used in the algorithms. However, RL researchers are paying increasing attention also to the safety of the approaches (e.g., avoiding falls, crashes, etc.) during the learning process (Geibel and Wysotzki, 2005; García and Fernández, 2015). Thus, when using RL techniques for humanoid walking, practical deployment of learning algorithms must contend with the fact that the training process itself may be unsafe for the robot. Then, an important question arises; namely, how can we ensure that the exploration of the state–action space will not cause damages or falls of the robot, while, at the same time, learning (near)optimal policies? The matter, in other words, is one of ensuring that the humanoid is able to explore a dangerous environment both safely and efficiently also during training. One could argue that such a training process may be performed first in simulation (where it is not necessary to behave safely) and, afterwards, transfer the learned safe policy from simulation to the real environment. This presents two drawbacks. On one hand, such simulators are not always available. On the other hand, behaviors learned in simulation are not always transferable to real environments due to a significant drop of their performance (Mouret and Chatzilygeroudis, 2017). Simulation cannot reproduce real conditions accurately enough, and, hence, a safe policy trained with simulation could have catastrophic consequences tested in the real environment. Therefore, it would be highly desirable to apply the learning algorithm directly to the real robot and, to that, two conditions are required: (i) fast convergence of the learning algorithm to (near)optimal policies, and (ii) safe exploration of the state and action space.

The main contribution of this paper is concerned with the application of a Safe RL algorithm, that meets the previous two requirements, to the task of humanoid robot walking. PI-SRL (García and Fernández, 2012) is our previous algorithm for safe exploration in dangerous and continuous control tasks. Such a method requires a predefined (and safe) baseline policy, which is assumed to be suboptimal (otherwise, learning would be pointless). PI-SRL is based in a risk function, that measures the risk of a state in terms of its similarity to previously visited known states in a case base. In that work, the risk function is defined as a binary step function. In contrast to this binary step risk function, one would expect that a continuously increasing monotonic risk function would provide a smoother transition between known

让人形机器人在训练的时候也可以保障安全

现在的模式基本上是先仿真，再迁移到现实中：但是仿真并不总是有的；而且有的话仿真有用也不代表现实有用。

基于12年的PI-SRL进行改进

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2019.103360.

* Corresponding author.
E-mail addresses: fjaviergp@gmail.com (J. García), diogoshafie@gmail.com (D. Shafie).

https://doi.org/10.1016/j.engappai.2019.103360
Received 14 April 2019; Received in revised form 1 November 2019; Accepted 4 November 2019
Available online 11 November 2019

and unknown states. Therefore, in this paper, we propose to use a continuously increasing monotonic risk function that determines the probability to follow the baseline policy, which results in a new algorithm, called ***P**olicy **R**euse for **S**afe **R**einforcement **L**earning* (PR-SRL). We use the PR-SRL algorithm to learn the optimal locomotion walking pattern in a humanoid robot while avoiding falling down, where velocity is the parameter to maximize. The algorithm uses a pre-defined configuration of a given walking behavior as baseline policy during the learning process. The experiments demonstrate that the robot learns to walk faster, while reducing to a minimum degree the number of falls during learning, when compared with state-of-the-art algorithms. Different approaches have been used to learn real robots tasks (Meriçli and Veloso, 2010; Kober and Peters, 2011; Wu et al., 2018), but, as far as we know, without an explicit mention of the risk concept, and without implementing automatic mechanisms to detect and to prevent risk situations.

Regarding the organization of the paper, Section 2 summarizes relevant related work. Section 3 introduces key concepts for RL. Section 4 describes the main concepts of Safe RL, including the PR-SRL algorithm used for biped walking. Section 5 summarizes the humanoid robot used to conduct the experimentation, the *NAO* robot (Gouaillier et al., 2009), and the mapping of the walking task onto a RL task. Section 6 reports the evaluation performed and, finally, Section 7 summarizes the main conclusions of this manuscript.

2. Related work

Designing and improving locomotion policies for biped walking is a complex task far from trivial, and it has been approached in many different ways using simulated or real environments. In simulation, Torres and Garrido (2011) uses the Webots mobile robot simulator (Webots, 2013) to simulate the Nao robot bipedal locomotion. They suggests the use of genetic algorithms to create a robot bipedal controller that is equivalent in the way humans and animals generate locomotion. Instead, Nikbin et al. (2011) use a Particle Swarm Optimization algorithm to find optimized walking parameters for a Simulated NAO robot of the RoboCup soccer simulation environment. More recent works in learning locomotion patterns for biped walking are those by Castro et al. (2017) and Gil et al. (2019). However, it is important to note that all these works demonstrate to obtain good policies in simulation environments (where it is not important to behave safely), but they does not transfer these policies to real robots in order to evaluate its performance and safety. Additionally, the authors do not explain the applicability of their approaches in real robots taking into account time restrictions (in real environments random exploration of the state space can get even more prohibitive due to reasons such as safety and/or large convergence time) or the need to prevent falls also during training.

In real robots, Meriçli and Veloso (2010) use a two phase biped walk learning approach based on learning from demonstration. Their method learns movement corrections to the walk based on the corrective feedback provided by a human, while walking autonomously using an analytical simplified walk algorithm. However, the human corrections proposed by Mereçli and Veloso depends on the human to identify the risk situation (i.e., the human must visually detect when the robot may be falling) and then makes the correction at the appropriate time, which may too late to avoid the fall. Our approach is not based on the responsiveness of a human to detect risk situations, it is based on the distance between the known space and unknown space. Farchy et al. (2013) propose an iterative optimization algorithm to walk faster. However, each iteration of the algorithm requires to learn the behavior on the simulator, test it on the real robot, modify the simulator to correct the imperfections, and so on; each iteration requires much time and the number of iterations that can be performed is limited. Lutz et al. (2012) uses kinesthetic teaching (Oßwald et al., 2011) that enable a humanoid robot to traverse ramps using only vision and inertial data for sensing. However, this approach is limited to the human choose the right pose at the appropriate time, and the choice of a wrong pose could make fall the robot. Shafii et al. (2010) uses Truncated Fourier Series (TFS) to generate angular trajectories for the robot joints. Then, Particle Swarm Optimization is used to find the best angular trajectories and optimize TFS. However, PSO also requires the random exploration of the solution space and such form of exploration can lead the agent to dangerous situations.

There has also been work on using RL to improve robot walking in real environments. Kulk and Welsh (2011) use Policy Gradient RL (Kohl and Stone, 2004) with a fitness function to walk faster and more stable in a physical Nao robot. However, conventional policy-gradient based approaches requires the use of a stochastic policy and a batch of episodes executing a random exploration to compute the gradient information, and this means a negative impact in the learning speed and in the safety of the system. To overcome the problem of the computation of the gradient information, Kober and Peters (2011) presented the PoWER algorithm, a policy search approach based on expectation–maximization. PoWER implement a different policy perturbation scheme, where the parameters of the policy, rather than its output (i.e., the actions to be performed in every state), are randomly perturbed. However, once again, such random exploration can lead the agent to dangerous situations, and PoWER has no mechanism to detect or avoid these situations. Current approaches based on Deep RL (Duan et al., 2016) and Evolutionary RL (Koppejan and Whiteson, 2011; Miikkulainen, 2017) for robot control present a similar problem. They are based on the random exploration of the state–action space or the policy space, without implementing any mechanism to avoid dangerous situations during such a exploration. Finally, it is also worth mentioning that Mescheder (2011) also use RL techniques to optimize the walking pattern in a real robot. In this work, the reward function returns a negative reward of −100 for falling over, but the approach does not introduce an explicit mechanism to avoid enter in this kind of situations from the beginning of the learning process.

Finally, Table 1 analyzes these RL approaches used in real robotic tasks across four dimensions: the mechanism they use for risk detection (column entitled *Risk detection* in Table 1), how they use the demonstrations provided from humans or automatic controllers (*Prior knowledge*), the exploration strategy they use for action selection (*Exploration*), and examples of real robotic tasks where these algorithms has been used (*Tasks*). Table 1 also analyzes our algorithm PR-SRL across these four dimensions in such a way that the reader can rapidly appreciate the contribution of our work to the state-of-the-art.

According to Table 1 we would like to highlight three main aspects. First, our algorithm PR-SRL is the only approach able to detect dangerous situations through the use of a risk function during training. In all other approaches, as far as we know, there are not mechanisms to detect such a risk situations. Second, PR-SRL is the only approach that use prior knowledge also during training. In all other cases, example demonstrations (from humans or automatic controllers) are provided to bootstrap the learning algorithms (i.e., as a type of initialization procedure), but not during training. However, as demonstrated in our previous work (García and Fernández, 2012), such initialization is not enough to avoid dangerous situations: it is necessary to provide such knowledge also during training. For this reason, during training PR-SRL is able to detect dangerous situations through the use of a risk function and use the baseline policy to return to safe states as described in Section 4. Therefore, thirdly, PR-SRL provides a guided and safe exploration towards the most promising regions of the space, while the rest of algorithms use a random exploration, so they are blind to the risk of actions, potentially ending up in catastrophic states as demonstrated in Section 6.

3. Background on Reinforcement Learning

A RL environment is typically formalized by means of an MDP (Sutton and Barto, 1998). An MDP consists of a set of states S, a set of

Table 1
This table lists most of the methods based on RL discussed in this related work and classifies each in terms of four dimensions.

Algorithm	Risk detection	Prior knowledge	Exploration	Tasks
Policy Gradient (Kohl and Stone, 2004; Kulk and Welsh, 2011; Kober and Peters, 2011)	–	Initialization	Random action perturbation	Biped walking, Quadrupedal locomotion, Motor trajectories
PoWER (Kober and Peters, 2011; Kormushev et al., 2013; Yuan et al., 2019)	–	Initialization	Random parameter perturbation	Biped walking, Motor trajectories
Deep RL (Duan et al., 2016; Gu et al., 2017; Kahn et al., 2018; Huang et al., 2019)	–	Initialization	Random action perturbation	Quadrupedal locomotion, Navigation, Manipulation
Evolutionary RL (Chernova and Veloso, 2004; Koppejan and Whiteson, 2011; Miikkulainen, 2017; Hirayama et al., 2017)	–	Initialization	Random parameter perturbation	Quadrupedal locomotion, Manipulation
Q-learning (Mescheder, 2011)	–	Initialization	Random action selection	Biped Walking
PR-SRL	Continuous risk function	Initialization/ Training	Guided action perturbation	–

actions A available from each state, the reward function $R : S \times A \to \mathbb{R}$ which assigns numerical rewards to transitions, and transition probabilities $T : S \times A \times S \to [0, 1]$ that capture the dynamics of a system. In this paper, we consider S and A to be infinitely large bounded sets, i.e., we consider MDPs where both the states and actions are continuous. We also consider discrete-time MDPs, i.e., at each discrete time step n the learning agent perceives a state $s \in S$ and takes an action $a \in A$ that leads it to the next discrete time step $n + 1$, with $n \in \{1, 2, 3, \dots\}$. The goal is to learn a policy π, which maps each state to an action, such that the return $J(\pi)$ is maximized:

$$J(\pi) = \sum_{h=0}^{H} \gamma^h r_h \tag{1}$$

where r_n is the immediate reward received in step n, and γ is the discount factor and affects how much the future is taken into account (with $0 \leq \gamma \leq 1$). We assume that the interaction between the learning agent and the environment is broken into episodes, where H is a time instant at which a terminal state is reached, or a fixed length for a finite horizon problem. Traditional methods in RL, such as TD-learning (Sutton and Barto, 1998), typically try to estimate the return (sum of rewards) for each state s when a particular policy π is being performed. This is also called the value-function $V^\pi(s) = E[J(\pi)|s_0 = s]$. The value of performing an action a in a state s under policy π is represented as $Q^\pi(s, a) = E[J(\pi)|s_0 = s, a_0 = a]$- this value represents the estimated return, i.e. sum of rewards, the system will receive when it performs action a in the state s, and follows the policy π thereafter. The Q-function is also called the action-value function.

To correctly approximate the value or the action-value function, RL learning algorithms use exploration/exploitation strategies (e.g., ϵ-greedy, softmax) as a balance between the exploration of random unexplored actions and the exploitation of the ongoing learned policy (Tijsma et al., 2016). However, typically these exploration strategies do not implement mechanisms to detect and to avoid risk situations as the strategy proposed in this paper.

4. Safe Reinforcement Learning

In this section, we describe the main elements of Safe Reinforcement Learning. A wider explanation of its basis, as well as previous algorithms, can be found in the literature (García and Fernández, 2012, 2015). As described previously, in this paper, we assume the presence of a baseline behavior, π_T, able to provide safe demonstrations of the biped walking task, and to advise suboptimal actions in unknown states to reduce the probability of entering into dangerous situations. That means that while the baseline behavior might not be able to indicate the best action in all cases, the action it supplies should, at the very least, be safer than that obtained through random exploration

By a way of introduction, the intuition behind the safe exploration used in this paper is as follows. At the beginning the learning agent knows nothing about the environment, i.e, for the learning agent, all the states are *unknown* states. We consider that an *unknown* state is an *unsafe* state because the agent does not know what action to perform on it. Certainly, not knowing what to do in a given situation increases the probability of ending up in trouble. Thus, in our case, the concept of *risk* is associated to the concept of *unknown* (García and Fernández, 2012). In case of an *unknown* state, the risk is maximum, hence, it increases the probability that the learning agent asks the baseline behavior, π_T, for advice. Then, this *unknown* state together with the action suggested by π_T in that state are stored by the learning agent, and the *unknown* state becomes a *known* state. In this way, if the agent revisits a state similar (or equal) to the new *known* state, it will know what action to perform on it. As the learning process proceeds, the learning agent gradually discovers more of the environment around it and, then, it begins to explore beyond the actions provided by π_T. To this end, it adds small amounts of Gaussian noise or perturbations to the actions previously provided by the baseline behavior in order to find new and better ways of completing the task. Therefore, although a baseline behavior is used, the objective is to explore beyond what is provided in the demonstrations by π_T. In some cases, such a exploration leads the learning agent to new *unknown* regions of the state space. The learning agent is assumed to be able to detect such situations with a risk function and, then, it asks for advice to the baseline behavior to return to safe, *known* states. The iteration of this process leads the learning agent to progressively and safely explore the state and action spaces in order to find new and improved ways to complete the task, while avoiding the visits to dangerous or *error* states.

It is important to be aware of the fact that this exploration is based on parent–child learning, where the baseline behavior, π_T, takes the role of the parent and the learner takes the role of the child. At the beginning, the child does not know much about the world, hence, asks the parent for advice. The child incorporates the parent's knowledge into his/her own knowledge about the world. However, the curiosity leads the child to explore new actions beyond the advice of the parent. Therefore, the child investigates new similar actions around the actions

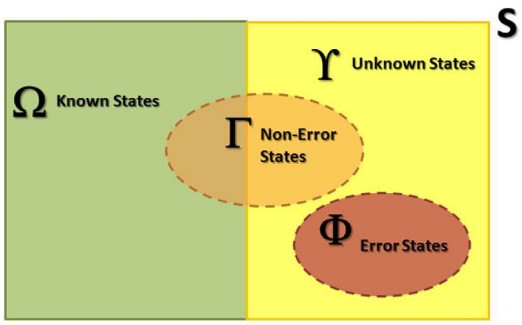

Fig. 1. Known/unknown and error/non-error states.

advised by the parent. In this process, he/she can discover new and better actions that the advised by the parent to perform the task. However, if during this exploration process for better actions the child is again in an *unknown* situation, he/she request immediately the advice of his/her parent. In this way, the child safely explores the world.

It is also important to note that this form of exploration requires dividing the state space into error/non-error and unknown/known states. Section 4.1 provides a formal description of this partition. It also requires the definition of a risk function able to detect risk situations (Section 4.2), and a safe exploration/exploitation strategy (Section 4.3). Finally, Section 4.4 present PR-SRL, the algorithm used to safely learn the biped walking task. This algorithm uses the π-reuse exploration strategy to safely balance the exploitation of actual learned knowledge, the exploration of new actions, and the request of teacher advice in considered dangerous parts of the state space.

4.1. Partitioning the state space: Error and non-error, and known and unknown states

In this paper, the state space is divided as described in Fig. 1. Regarding error/non-error states, we follow as far we can the notation presented in Geibel and Wysotzki (2005) for the definition of error and non-error states. In their study, Geibel and Wysotzki (2005) associate risk with error states and non-error states, with the former understood as a state in which it is considered undesirable or dangerous to enter.

Definition 1 (*Error and Non-error States*)**.** Let S be a set of states and $\Phi \subset S$ the set of error states. A state $s \in \Phi$ is an undesirable terminal state where the control of the agent ends when s is reached with damage or injury to the agent, the learning system or any external entities. The set $\Gamma \subset S$ is considered a set of non-error terminal states with $\Gamma \cap \Phi = \emptyset$ and where the control of the agent ends normally without damage or injury.

In terms of RL, if the agent enters an error state, the current episode ends with damage to the learning system (or other systems); whereas if it enters a non-error state, the episode ends normally and without damage.

In respect to the unknown/known states, let us assume the learning agent is equipped with a case-base $B = \{c_1, \ldots, c_\eta\}$. In B, every case c_i consists of a state–action pair (s_i, a_i) that the agent has experienced in the past and with an associated value $V(s_i)$. Thus, $c_i = \langle s_i, a_i, V(s_i)\rangle$, where the first element represents the case's problem part and corresponds to the state s_i, the following element, a_i, depicts the case solution (i.e., the action expected when the agent is in the state s_i) and the final element, $V(s_i)$, is the value function associated with the state s_i. Each state s_i is composed of n continuous state variables and each action a_i is composed of m continuous action variables.

Hence, the cases in B describe a **Case Based Policy** of the agent, π_B^θ, and its associated value function $V^{\pi_B^\theta}$. When the agent receives a new state s_q, the agent first retrieves the nearest neighbor to the s_q point in B according to some similarity metric and then the associated action is performed. In this paper, we consider the Euclidean distance as similarity metric Eq. (2).

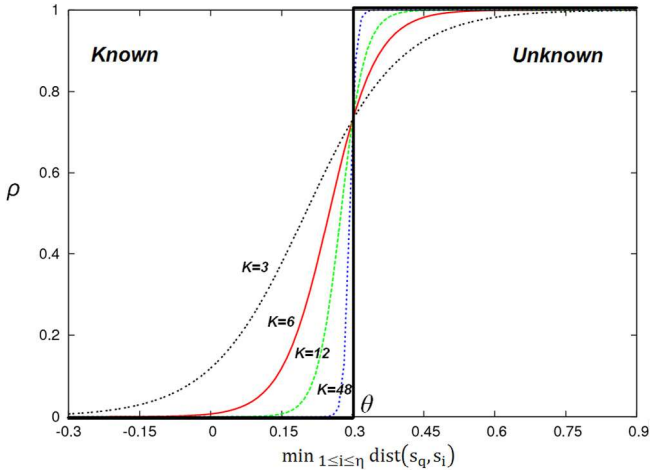

Fig. 2. Binary step function, and continuous risk function for different values of k. The parameter θ is set to $\theta = 0.3$.

$$d(s_q, s_i) = \sqrt{\sum_{j=0}^{n} (s_{q,j} - s_{i,j})^2} \tag{2}$$

A *density threshold*, θ, is used to determine when a new case should be added to the memory. When the distance of the nearest neighbor to s_q is greater than θ, a new case is added to the memory. In this sense, the parameter θ defines the size of the classification region for each case in B.

Definition 2 (*Known and Unknown States*)**.** Given a case base $B = \{c_1, \ldots, c_\eta\}$ composed of cases $c_i = (s_i, a_i, V(s_i))$, a state s_q is considered **known**, when $\min_{1 \le i \le \eta} d(s_q, s_i) \le \theta$; the state s_q is considered **unknown** otherwise. Formally, we consider a set $\Omega \subseteq S$ of **known** states, and we allow an additional set of **unknown** states $\Upsilon \subseteq S$ with $\Omega \cap \Upsilon = \emptyset$, and $\Omega \cup \Upsilon = S$.

When the agent receives a new state $s \in \Omega$, it performs the action a_i of the case c_i for which $d(s, s_i) = \min_{1 \le j \le \eta} d(s, s_j)$ (known state). However, if the agent receives a state $s \in \Upsilon$ where, by definition, the distance to any state in B is larger than θ (unknown state), no case is retrieved. Consequently, the action to be performed from that state is unknown to the agent. It is important to bear in mind that at the beginning of the exploration process, the agent does not have any information about the environment, hence, all the states are unknown, i.e., at the beginning of the exploration process it is $\Upsilon = S$.

4.2. The risk function

In this paper, we use a continuous risk function in order to *measure* the quantity of risk of a given state s. Given a case base $B = \{c_1, \ldots, c_\eta\}$ composed of cases $c_i = (s_i, a_i, V(s_i))$, the risk for each state s is defined as Eq. (3).

$$\varrho^B(s) = 1 - \frac{1}{1 + e^{\frac{k}{\theta}((\min_{1 \le j \le \eta} d(s, s_j)) - \frac{\theta}{k}) - \theta)}} \tag{3}$$

Eq. (3) allows us to obtain a smoother transition between risk-free states (i.e., known states) and risk states (i.e., unknown states) as described in Fig. 2.

The parameter k has a double effect. On the one hand, depending on its value, the width of the sigmoid function varies. A lower value of k implies a wider sigmoidal function. On the left of the parameter θ

in Fig. 2, this implies a higher probability of consider *known* states as *unknown* states This results in a less aggressive exploration of the state space during the learning process since the baseline behavior advices are more frequently required (being able to affect negatively the final performance of the algorithm). On the other hand, the parameter k is used to displace the sigmoid function to the left, reducing in this way the probability of consider *unknown* states as *known* states. However, it is important to note that this probability does not disappear completely, i.e., this displacement allows also to keep the smooth transition to the right of the θ parameter (Fig. 2). In summary, using Eq. (3), lower values of k imply a less aggressive exploration of the state space, and a lower probability of consider unknown states as known. Instead, the higher the values of k are, the more similar the continuous risk functions in Fig. 2 will be with the binary step function. This implies a more aggressive exploration of the state space, increasing the probabilities of damages.

The risk function in Eq. (3) can be seen as the probability that a state will be considered as an *unknown* state. Therefore, from Eq. (3), the application of probabilistic policy reuse to measure the advice of a teacher in safe RL is easy, by using the risk function $\varrho^B(s)$ as a transfer function. The transfer rate depends on the safety of the learning agents: if safety is high (i.e., $\varrho^B(s)$ is low), probability to use the advice of the teacher is very low, while if safety is low (i.e., $\varrho^B(s)$ is high), such probability increases. This integration will be explained deeply in Section 4.4.

4.3. Safe π-reuse exploration/exploitation strategy

Algorithm 1 shows Safe π-reuse, a version of the π-reuse (Fernández and Veloso, 2006; Fernández et al., 2010) algorithm to incorporate the teacher advice in the exploration process. The main elements over the original π-reuse strategy are:

- the past policy Π_{past} is replaced by the baseline behavior π_T
- the new policy to be learned Π_{new} is replaced by the case base policy π_B^θ
- the parameter ψ is replaced by $\varrho^B(s)$
- no ϵ-greedy strategy is used, because actions are continuous. Instead, random Gaussian noise is used to generate exploratory actions.

Algorithm 1: Safe π-reuse $(\pi_T, H, B, \sigma, \theta, k)$

Input: π_T, H, B, σ, θ, k
Output: $listCasesEpisode$, $totalRwEpisode$

1 Initialize $listCasesEpisode \leftarrow \emptyset$, $totalRwEpisode \leftarrow 0$, $h \leftarrow 1$, initial state, s_h
2 **repeat**
3 Compute the case $< s, a, V(s) > \in B$ closest to the current state s_h;
4 $\varrho^B(s_h) \leftarrow 1 - \frac{1}{1+e^{\frac{k}{\theta}((\min_{1\le j\le \eta} d(s_h, s_j) - \frac{\theta}{k}) - \theta)}}$;
5 With a probability of $\varrho^B(s_h)$: $a_h \leftarrow \pi_T(s_h)$, $c^{new} \leftarrow (s_h, a_h, 0)$;
6 With a probability of $1 - \varrho^B(s_h)$: $a_h \leftarrow rnd\_gauss(\pi_B(s_h), \sigma)$, $c^{new} \leftarrow (s, a_h, V(s))$;
7 Execute a_h and receive the next state s'_h, and reward, r_{s_h,a_h};
8 $totalRwEpisode \leftarrow totalRwEpisode + r(s_h, a_h)$;
9 $listCasesEpisode \leftarrow listCasesEpisode \cup c^{new}$;
10 $s_h \leftarrow s'_h$;
11 $h \leftarrow h + 1$;
12 **until** $h < H$;

The Safe $\pi - reuse$ exploration/exploitation strategy is as follows. The algorithm builds a case for each step of an episode. For each new state s_h, the closest case $\langle s, a, V(s)\rangle \in B$ is computed using the Euclidean distance metric defined in Eq. (2) (see line 3 in Algorithm 1). At this point, the π-reuse strategy is followed, using the $\varrho^B(s)$ function as a transfer probability: with a probability of $\varrho^B(s)$ the policy of the baseline behavior π_T is followed, while with a probability of $1 - \varrho^B(s)$, the action suggested by current base case policy is executed. Therefore, in areas far from the known states, the probability to use the baseline behavior advice is very high, while this advice is rarely used in known areas. If the algorithm follows the baseline behavior the action a_h performed is suggested by the baseline behavior π_T which defines safe behavior, and a new case $\langle s_h, a_h, 0\rangle$ is built (line 5). In this case, the state s_h is considered an unknown state.

If the algorithm exploits the new policy the action a_h is performed (line 6). In this case, the new case $\langle s, a_h, V(s)\rangle$ is built replacing the action a corresponding to the closest case in $\langle s, a, V(s)\rangle \in B$, with the new action a_h resulting from the application of random Gaussian noise to a. In order to maximize exploration safety, it seems advisable that movement through the state space is not arbitrary, but rather that known space be expanded only gradually by starting from a known state. Therefore, in line 6 the action performed is sampled from a Gaussian distribution with the mean at the action output given by the case selected in B. The shape of the Gaussian distribution depends on parameter σ (standard deviation). In this study, σ is used as a *width parameter*. While large σ values imply a wide bell-shaped distribution, increasing the probability of selecting actions a_h very different from the current action a, a small σ value implies a narrow bell-shaped distribution, increasing the probability of selecting actions a_h very similar to the current action a. Finally, the reward obtained in the episode is accumulated, where $r(s_h, a_h)$ is the immediate reward obtained when action a_h is performed in state s_h (line 8) and the new case is added to the list of cases (line 9).

Fig. 3 graphically represents a running example of the lines 3–6 of Algorithm 1 for action selection, which are the heart of the algorithm. In the proposed example, the state $s_h = (1, 2)$ is perceived from the environment. First, then, it is computed the Euclidean distances between the perceived state $s_h = (1, 2)$ and each state s_i of the cases $c_i \in B$. Let us assume that the minimum distance is with the state $s_1 = (2, 3)$ corresponding to the case $c_1 = \langle(2, 3), 4, 8\rangle \in B$ and, thus, c_1 is the closest case to $s_h = (1, 2)$.

Once the closest case $c_1 \in B$ is selected, it is computed the risk function, $\varrho^B(s_h)$, using the distance $d(s_h, s_1)$ as described in Eq. (3). In the running example in Fig. 3, let us consider the risk function is $\varrho^B(s_h) = 0.7$. This value, 0.7, is used as a probability of considering the state s_h as a dangerous state. If that is the case, the action a_h to be performed in the next step is computed using the baseline behavior, $a_h = \pi_T(s_h)$. Otherwise, the action a_h is computed adding a small amount of Gaussian noise to the action in the closest case.

4.4. PR-SRL algorithm

The Safe π-reuse is used by the PR-SRL algorithm to conduct a safe exploration of the state and action spaces. The PR-SRL algorithm[1] used for biped walking is depicted in Algorithm 2. The algorithm is composed of three steps performed in each episode.

- **(a) Case generation**. The algorithm uses the π-reuse exploratory strategy in Algorithm 1 to build the cases in the episode (line 3 in Algorithm 2.

- **(b) Computing the state-value function for the unknown states**. In this step, the state-value function of the states considered to be unknown in the π-reuse exploration is computed. In the previous step (line 3), the state-value function for these states is set to 0. The algorithm proceeds in a manner similar to the first-visit MC algorithm (Sutton and Barto, 1998). In this case, the return for each considered unknown state s_i is computed, but not averaged since only one episode is considered (lines 6–7).

[1] The source code of PI-SRL and PR-SRL is available in https://bitbucket.org/fjaviergp/srl.

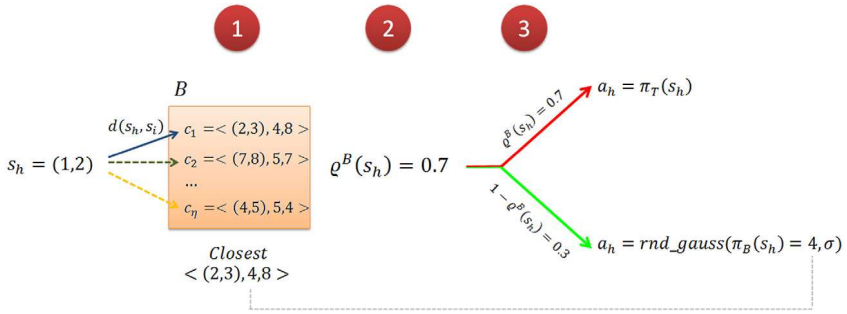

Fig. 3. Running example of the Safe π-reuse exploration/exploitation strategy for a given state $s_h = (1, 2)$. First, it is computed the closest case, $c_1 = \langle (2, 3), 4, 8 \rangle$, from B. Second, it is computed the risk function $\varrho^B(s_h)$. Finally, thirdly, it is computed the action a_h according to the probability given by $\varrho^B(s_h)$.

Algorithm 2: PR-SRL($\eta, \pi_T, \Theta, \sigma, \theta, k, P, H$)

Input: η,π_T,Θ,σ, θ, P, H
Output: B

1 Initialize *maxTotalRwEpisode* $\leftarrow 0$, $B \leftarrow \emptyset$;
2 **repeat**
/* (a) Case generation */
3 $listCasesEpisode, totalRwEpisode \leftarrow \pi$-reuse($\pi_T$,$H$,$B$,$\sigma$,$\theta$,$k$);
/* (b) Computing the state-value function for the unknown states */
4 **foreach** $c_i \in listCasesEpisode$ **do**
5 **if** s_i *was considered an* unknown state **then**
6 $return(s_i) \leftarrow \sum_{j=n}^{k} \gamma^{j-n} r(s_j, a_j)$;
7 $V(s_i) \leftarrow return(s_i)$;
8 **end**
9 **end**
/* (c) Updating the cases in B using the gathered experience */
10 **if** $totalRwEpisode > (maxTotalRwEpisode - \Theta)$ **then**
11 $maxTotalRwEpisode \leftarrow$ ***max***$(maxTotalRwEpisode, totalRwEpisode)$;
12 **foreach** $c_i =< s_i, a_i, V(s_i) > \in listCasesEpisode$ **do**
13 **if** s_i *was considered a* known state **then**
14 Compute the case $< s_i, a, V(s_i) > \in B$ corresponding to the state s_i;
15 Compute $\delta \leftarrow r(s_i, a_i) + \gamma V(s_{i+1}) - V(s_i)$;
16 **if** $\delta > 0$ **then**
17 Replace $< s_i, a, V(s_i) > \in B$ with $< s_i, a_i, V(s_i) > \in listCasesEpisode$;
18 $V(s_i) \leftarrow V(s_i) + \alpha\delta$;
19 **end**
20 **end**
21 **else**
22 $B \leftarrow B \cup c_i$;
23 **end**
24 **end**
25 **end**
26 **if** $\|B\| > \eta$ **then**
27 Remove the $\eta - \|B\|$ least-frequently-used cases in B;
28 **end**
29 $p \leftarrow p + 1$
30 **until** $p < P$;

- **(c) Updating the cases in B using gathered experience**. Updates in B are made with the cases gathered from episodes with a cumulative reward similar to that of the best episode found to that point using the threshold Θ (line 10). Thus, the updates in B will be made with the cases gathered from episodes with a similar quality to the best episode found so far. In this step, two types of updates appear, namely, replacements and additions of new cases. Again, the algorithm iterates for each case $c_i = (s_i, a_i, V(s_i)) \in listCasesEpisode$ (line 12). If s_i was considered a known state during the π-reuse exploration (line 13), we compute the case $\langle s_i, a, V(s_i) \rangle \in B$ corresponding to the state s_i (line 14). One should note that the case $c_i = (s_i, a_i, V(s_i)) \in listCasesEpisode$ was built in line 08 of Algorithm 1, replacing the action a corresponding to the case $\langle s_i, a, V(s_i) \rangle \in B$ with the new action a_i and resulting from the application of random Gaussian noise to the action a. Then, the temporal distance (TD) error δ is computed (line 15). If $\delta > 0$, performing the action a_i results in a positive change for the value of a state, and it is reinforced. In the algorithm, this reinforcement is carried out by updating the output of the case $\langle s_i, a, V(s_i) \rangle \in B$ at a_i (line 17). If, instead, s_i was considered a known state, the case c_i is added to B (line 22). Finally, the algorithm removes cases from B if necessary (line 27).

5. Robotic framework

In this section, we firstly describe the humanoid robot used to conduct the experiments (Section 5.1), and, we present the mapping of the biped walking task onto an episodic RL task for the application of Safe RL (Section 5.2).

5.1. NAO robot

In this paper, we use the Aldebaran NAO V5 robot to conduct the experiments (Fig. 4). It weighs 4.5 kg, is 57 cm high and has 22 degrees of freedom (DoFs).

NAO V5 has an on-board ATOM Z530 1.6 GHz processor, 1 GB RAM and 2 GB Flash memory, to be shared between the low level control system and the autonomous perception, cognition, and motion algorithms. It is equipped with several sensors including two color cameras, two ultrasound distance sensors, a 3-axis accelerometer, a 2-axis gyroscope (X–Y), an inertial measurement unit for computing the absolute orientation of the torso, 4 pressure sensors on the sole of each foot, and a bump sensor at the tip toe of each foot (Gouaillier et al., 2009).

5.2. Mapping humanoid walking onto a reinforcement learning task

In our biped walking problem, the state and action spaces are continuous. The state space S is represented by a 14-dimensional vector that reflects the configuration of the robot. It is composed of the values of 8 joint angles, the readings from the accelerometer in the x, y and z dimensions, and the readings from the gyroscope in the x, y and z dimensions. Similarly, the action space A is a 8-dimensional vector representing the torque directly applied to that 8 joints. Therefore, the action vector $a \in (-1, 1)^8$ specifies the fraction of maximum torque to be used in positive or negative direction. However, the implementation of an action in this form was not straightforward as NAO has a built-in feedback control mechanism that maintains specified angles of robot's

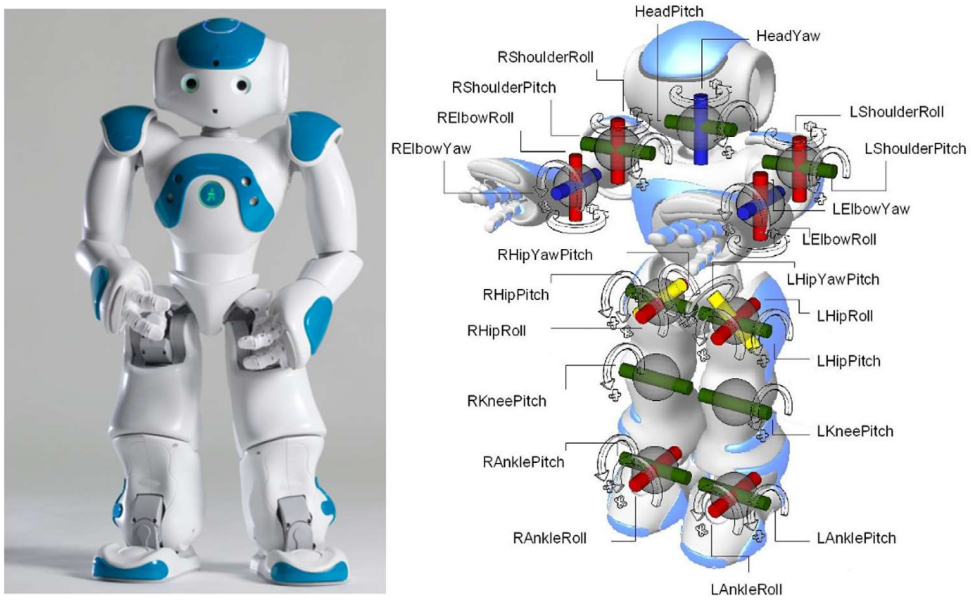

Fig. 4. Nao robot.

joints. The API allows to set new posture to be maintained and a torque limit. To simulate direct torque control, we set the torque limit for every joint and compute the change in the maintained angle of ith joint $\Delta\alpha_i$ by following relationship in Eq. (4).

$$\Delta\alpha_i = \omega_i * a_i * \Delta t \tag{4}$$

where ω_i is the maximum rotation velocity of the ith joint (with full torque), and Δt is the duration of the action. Our approach to allow the algorithm to access the robot hardware, like many other researchers (Ashar et al., 2015; Röfer and Laue, 2014; Bahdi, 2018), is through the *Device Communication Manager* (DCM). This module provides the fastest way to control the robot and, thus, reduces latency problems: it allows sending commands to the actuators and a rapid updating of actuator/sensor values.

The joints considered, which are the same for the left and right side of the robot, are: shoulder pitch, hip roll, hip pitch, and knee pitch. While the robot is walking, the feet are supposed to be parallel to the ground. This is achieved by using the ankle joints. Therefore, in this case, to learn the biped walking behavior, there are 8 DoFs. We tackled this biped robot walking as an episodic task, where each episode ends when the robot falls down or reaches a maximum length duration (we have limited the maximum time the robot can be walking). Each time step in the episode spans 200 ms, and the maximum number of steps of an episode is fixed to 100. We consider the objective of maximizing the distance traveled, so the reward function r is computed considering the distance that the robot has traveled in a single time step. Therefore, it is important to keep in mind that since the episodes have a maximum duration, and the objective is to maximize the distance traveled by the robot in these episodes, indirectly we are encouraging the robot to walk as fast as possible

6. Experimental results

This section presents the experimental results collected from the use of PR-SRL for policy learning in the real Aldebaran NAO robot described in Section 5.

6.1. Experimental scope

In this domain, the experiments have been conducted in order to learn a near-optimal policy which maximizes the traveled distance of NAO, and minimizes the number of falls during learning. During the experiments, the robot is equipped with a harness managed by a human that prevents the robot falls completely to the ground, thus, prevent possible hardware damage. A similar harness is used by Meriçli and Veloso (2010). It is important to note that the harness is used only to prevent the robot from hitting the ground, but not to guide its steps. Obviously, the ideal is that the learning processes could be conducted without the need of these external devices, since they require the rapid reaction of the human to avoid falling, which is not always the case. The experiments in Section 6.2 demonstrate that PR-SRL does not need these devices while other algorithms do need. In each episode, NAO begins from the same initial position and pose. Therefore, at the end of each episode, regardless of the distance traveled, NAO is transported again to this initial position, and adjusted to the start pose.

The results of PR-SRL in this domain are compared to those yielded by four additional techniques namely: TFS-PSO which is based on the use of Truncated Fourier Series (TFS) to model the angular trajectories of the joints, and on the use of Particle Swarm Optimization (PSO) to optimize its parameters (Shafii et al., 2010), HCD which is a biped walking learning approach based on human corrective demonstrations (Meriçli and Veloso, 2010), the PoWER algorithm in which the policy is parameterized using *Dynamic Motor Primitives* (DMPs) (Kober and Peters, 2011), and DDPG which is a successful RL algorithm based on *Deep Learning* and *Policy Gradient* (Wu et al., 2018). It is important to note that in this paper TFS-PSO, HCD, PoWER and DDPG do not begin learning from scratch since they are initialized using exactly the same baseline behavior π_T as that used in our PR-SRL algorithm. In the case of TFS-PSO in which a population of particles/solutions are required at the beginning of learning, each of the particles are generated by slightly perturbing π_T. This makes the comparison of performances as fair as possible. However, it is important to be aware of the fact that we are not trying to prove whether our algorithm is better than TFS-PSO, HCD, PoWER, or DDPG since they make its own use of its exploration/exploitation strategies. Instead we will take the results achieved with all these algorithms as reference to analyze the performance and the number of robot falls achieved with our algorithm. Additionally, the performance of PR-SRL is also compared with that of PI-SRL (García and Fernández, 2012), our previous algorithm which makes use of a step risk function instead of a continuous risk function as described in Section 4.2.

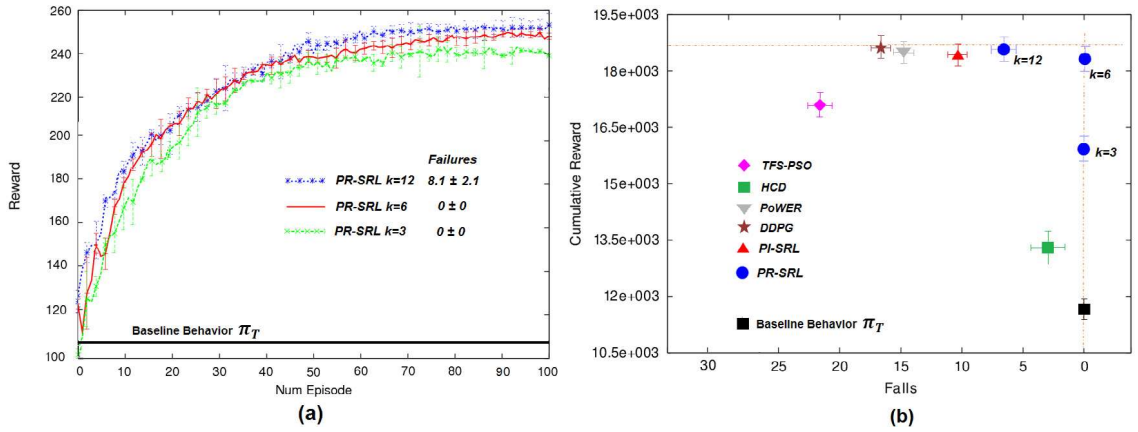

Fig. 5. (a) Mean cumulative reward per episode obtained by PR-SRL using different values of k. (b) Mean number of falls and cumulative reward obtained by PR-SRL using different values of k. The means and standard deviations have been computed from 10 different runs.

6.2. NAO results

Fig. 5(a) shows three different learning processes corresponding to PR-SRL using different values of k: $k = 3$, $k = 6$ and $k = 12$. In Fig. 5(a), we have run the algorithm 10 times due to the stochastic nature of the learning process. Therefore, Fig. 5(a) shows the average results together with their standard deviations. The conclusion to be drawn from Fig. 5(a) is that small k values perform a less aggressive exploration of the state and action space which slightly affect the final performance of the algorithm, although the obtained behavior is safer. Instead, high k values perform a more aggressive exploration, which also implies a higher probability of failure. In PR-SRL $k = 12$ the continuous risk function begins to look like the binary step function, and the failures appear. However, PR-SRL $k = 6$ is able to completely avoid the falls, while maintaining a similar performance achieved by PR-SRL $k = 12$. It is also important to note that PR-SRL clearly exceeds the performance of the baseline behavior π_T used. Finally, Fig. 5(a) may also be used to analyze the learning speed of the proposed algorithm. The learning curves in Fig. 5(a) demonstrate that PR-SRL reaches convergence at around episode 70. From this episode on, the learning speed for all the learning processes decreases, and the performance of the ongoing policies that are being learned remains constant. Therefore, our algorithm needs at around 70 episodes to learn this task. Obviously, the baseline behavior biases the learning algorithm towards promising regions of the search space, so as to reduce the learning time.

Fig. 5(b) shows the mean number of falls and cumulative reward over 100 episodes for different approaches. The data has been computed from 10 independent executions of each approach. In particular, Fig. 5(b) shows the performance of TFS-PSO (pink diamond), the performance of HCD (green square), the performance for PoWER (inverted gray triangle), the performance of DDPG (brown star), PI-SRL (red triangle) and, finally, PR-SRL with different k values incrementally tested: $k = 3$, $k = 6$ and $k = 12$ (blue circles). Fig. 5(b) shows that PR-SRL $k = 6$ is the one that has the best balance between cumulative reward obtained and number of falls. It reaches a similar performance that the best performances reached by PoWER and DDPG, but unlike them, it is able to completely avoid falls. This demonstrate that PR-SRL $k = 6$ is able to conduct a safe exploration of the state and action space while learn near-optimal policies. PR-SRL is also able to beat our previous algorithm PI-SRL, also demonstrating that the use of a continuous risk function is better than a binary step one, since it allows a smoother transition between *known* and *unknown* states.

Fig. 6 shows the walking speed of the policies obtained for each algorithm at the end of the learning processes.

Fig. 6 is a box plot showing the minimum, first quartile, median, third quartile, and maximum of the speed obtained by each policy from 10 different runs. As can be seen, PR-SRL $k = 6$ obtains a policy

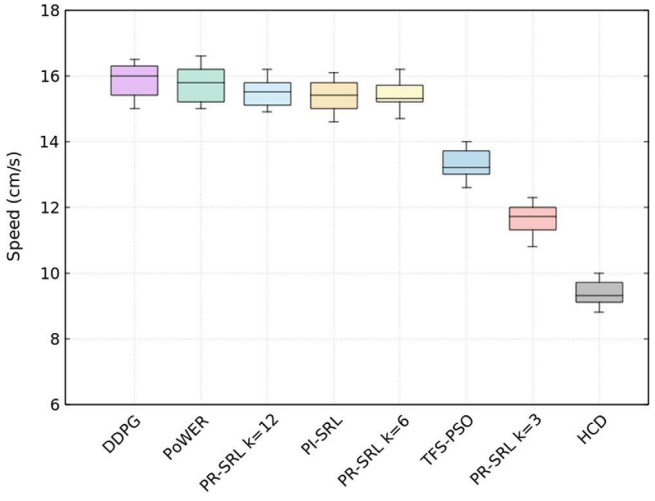

Fig. 6. Walking speed (cm/s) of NAO using the policies obtained for each algorithm at the end of the learning processes. The minimum, first quartile, median, third quartile, and maximum for each algorithm have been computed from 10 different runs.

with a high speed similar to the best ones and, additionally, unlike the others, it learns this policy without falls as demonstrated in Fig. 5(b). It is important to bear in mind therefore that the problem we are considering is multi-objective: it is not only about learning to walk as fast as possible, but about learning to walk as fast as possible without damage and, in this case, PR-SRL $k = 6$ outperforms all others.

Fig. 7 shows a view-from-above of the NAO robot and the trajectories produced by the baseline behavior, and the policies during the learning of PR-SRL $k = 6$. The red point in Fig. 7 shows the initial position of the robot. Fig. 7(a) shows the trajectories of the baseline behavior π_T. As can be seen the baseline behavior π_T in Fig. 7(a) do not produce falls, but it walk a short distance. In contrast, Fig. 7(b) shows the robot walk short distances following curved trajectories at the beginning, but, as the learning process proceeds, the traveled distances are getting longer and straighter. At the end of the learning process, the final policy learned by PR-SRL clearly outperform the distance traveled by the baseline behavior in Fig. 7(a).

As a final remark, Fig. 8 can be used to analyze the distance traveled by NAO at the end of the learning process using PR-SRL $k = 6$. It shows four snapshots taken at different time stamps ($t = 0$, $t = 3$, $t = 6$, and $t = 9$ s.), where the first snapshot corresponds to the initial pose of the robot before it begins to walk.

Additionally, for each time stamp, Fig. 8 shows a red dashed line to show the distance traveled by the baseline behavior at that same

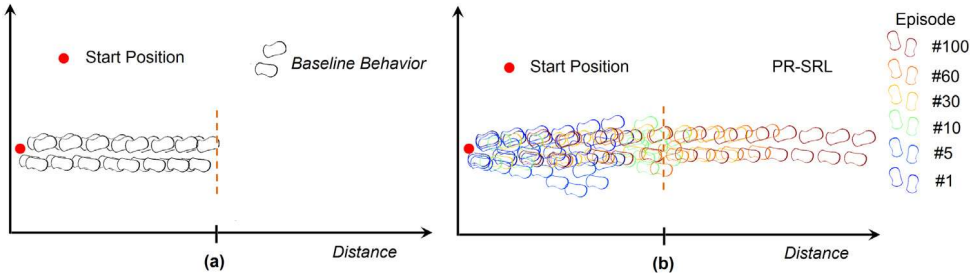

Fig. 7. Sample trajectories of (a) the baseline behavior π_T and (b) the policies obtained during the learning with PR-SRL $k = 6$.

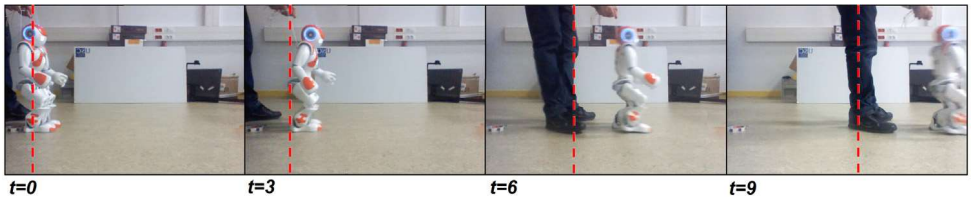

Fig. 8. Snapshots from NAO at four different time stamps ($t = 0$, $t = 3$, $t = 6$, and $t = 9$ s).

time. The final snapshot show that at the end of the learning process, the resulted policy greatly exceed the distance traveled by this baseline behavior, and it is completely stable and free of falls.

7. Conclusions

This paper describes the application of a Safe RL algorithm, PR-SRL, to the task of robot biped walking. The performance of the algorithm has been compared with state-of-the-art algorithms. Next we summarize the main conclusions found in this paper:

(i) Fast convergence and safe exploration of the state and action space. Typical RL methods perform a random exploration of the state and action space and this negatively affects both the learning speed and the safety of the learning system. However, learning speed and safety are two requirements particularly interesting, even more so we talk about robotic tasks. In this paper, PR-SRL has shown that it can be applied successfully to a robotic task as complex as biped robot walking. Section 6 demonstrates that PR-SRL maximizes the distance traveled by the NAO robot in a small number of episodes, while minimizing the number of falls when compared with other state-of-the-art approaches. In fact, the experiments in Section 6.2 demonstrated that PR-SRL $k = 6$ is able to learn near-optimal policies, without falling once.

(ii) It is impossible to completely avoid risk situations without certain prior knowledge. Some prior knowledge about the task is necessary to avoid risk situations. In our case, such a prior knowledge is given by the presence of a baseline behavior that safely demonstrates the task to be learned. PR-SRL uses such a baseline behavior for two purposes. On the one hand, it allows to bootstrap the learning algorithm (i.e., a sort of initialization procedure), and, on the other hand, to support the subsequent exploration process. Such subsequent exploration allows PR-SRL to go beyond the performance of baseline behavior as demonstrated in Section 6. However, it is important to note, nevertheless, that the presence of such a baseline behavior is not guaranteed in all domains and this limits the applicability of PR-SRL.

(iii) Safe RL to overcome the gap between simulation and reality. When deal with real robots, a typical approach is to learn first the behavior in a simulator, and then applying this learned behavior to the real robot. In a simulator it does not matter if the robot falls, since it will not suffer any real damage. Such approach would be ideal if simulators perfectly simulated reality, but this is not the case in most robotic tasks. In fact, many robotic platforms lack even of simulators. It is necessary, therefore, to build algorithms that can be applied directly in real robots, and it is imperative that these algorithms incorporate mechanisms to detect and avoid risk situations. This is the case of PR-SRL.

(iv) Safe RL to extend life-long of robots. The use of PR-SRL (or other risk sensitives approaches) for learning in real environments could reduce the amount of damage incurred and, consequently, allow the lifespan of the robots to be extended.

(v) As far as we know, this is the first time Safe RL is applied to a real robotic task. Different approaches have been used to learn in real robots as described in Section 2, but, as far as we know, without an explicit mention of the risk concept, and without implementing mechanisms to detect or to avoid risk situations. However, it is demonstrated in this paper that such mechanisms are essential when dealing with real robotic tasks. Obviously, learning directly in robots has other problems, such as power or latency problems that depend on the particular robotic platform to be used. It is important to bear in mind that our proposed algorithm is robot and task independent, but its application is limited to robotic platforms and tasks where these problems are not present or can be solved in some way.

In summary, PR-SRL is an interesting, and promising exploration approach for learning in real robots where it is mandatory to avoid (or at least minimize) the number of dangerous situations. One future work would be the deployment of the algorithm in other real robotic tasks, and its comparison with state-of-the-art algorithms.

Acknowledgments

Javier García developed this work during a postdoctoral stay at the University of Porto, being also partially funded by a postdoctoral grant under the UC3M-based research program, Spain.

References

Ashar, Jayen, Ashmore, Jaiden, Hall, Brad, Harris, Sean, Hengst, Bernhard, Liu, Roger, Mei (Jacky), Zijie, Pagnucco, Maurice, Roy, Ritwik, Sammut, Claude, Sushkov, Oleg, Teh, Belinda, Tsekouras, Luke, 2015. Robocup SPL 2014 champion team paper. In: Bianchi, Reinaldo A.C., Akin, H. Levent, Ramamoorthy, Subramanian, Sugiura, Komei (Eds.), RoboCup 2014: Robot World Cup XVIII. Springer International Publishing, Cham, pp. 70–81.

Bahdi, Emile, 2018. Development of a Locomotion and BalancingStrategy for Humanoid Robots (PhD thesis). Engineering and Computer Science, University of Denver, USA.

Castro, Guilherme Barros, Tamura, Kazuya, Kawamura, Atsuo, Hirakawa, André R., 2017. Biologically-inspired neural network for walking stabilization of humanoid robots. In: Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017., pp. 96–104.

Chernova, Sonia, Veloso, Manuela, An evolutionary approach to gait learning for four-legged robots. In: Proceedings of IROS'04.

Duan, Yan, Chen, Xi, Houthooft, Rein, Schulman, John, Abbeel, Pieter, 2016. Benchmarking deep reinforcement learning for continuous control. In: Balcan, Maria Florina, Weinberger, Kilian Q. (Eds.), Proceedings of the 33rd International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 48, PMLR, New York, New York, USA, pp. 1329–1338.

Farchy, Alon, Barrett, Samuel, MacAlpine, Patrick, Stone, Peter, 2013. Humanoid robots learning to walk faster: From the real world to simulation and back. In: Proc. of 12th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS).

Fernández, Fernando, García, Javier, Veloso, Manuela, 2010. Probabilistic policy reuse for inter-task transfer learning. Robot. Auton. Syst. 58(7), 866–871.

Fernández, Fernando, Veloso, Manuela, 2006. Probabilistic policy reuse in a reinforcement learning agent. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. In: AAMAS '06, ACM, New York, NY, USA, pp. 720–727.

García, Javier, Fernández, Fernando, 2012. Safe exploration of state and action spaces in reinforcement learning. J. Artificial Intelligence Res. 45, 515–564.

García, Javier, Fernández, Fernando, 2015. A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. 16 (1), 1437–1480.

Geibel, Peter, Wysotzki, Fritz, 2005. Risk-sensitive reinforcement learning applied to control under constraints. J. Artif. Intell. Res. (JAIR) 24, 81–108.

Gil, Cristyan R., Calvo, Hiram, Sossa, Humberto, 2019. Learning an efficient gait cycle of a biped robot based on reinforcement learning and artificial neural networks. Appl. Sci. 9 (3).

Gouaillier, David, Hugel, Vincent, Blazevic, Pierre, Kilner, Chris, Monceaux, Jérôme, Lafourcade, Pascal, Marnier, Brice, Serre, Julien, Maisonnier, Bruno, 2009. Mechatronic design of nao humanoid. pp. 769–774,

Gu, Shixiang, Holly, Ethan, Lillicrap, Timothy, Levine, Sergey, 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: Proceedings 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, Piscataway, NJ, USA.

Hirayama, Chiaki, Watanabe, Toshiya, Kawabata, Shinji, Suganuma, Masanori, Nagao, Tomoharu, 2017. Acquiring grasp strategies for a multifingered robot hand using evolutionary algorithms. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017, Banff, AB, Canada, October 5-8, 2017. pp. 1597–1602.

Huang, Sandy H, Zambelli, Martina, Kay, Jackie, Martins, Murilo F, Tassa, Yuval, Pilarski, Patrick M, Hadsell, Raia, 2019. Learning gentle object manipulation with curiosity-driven deep reinforcement learning. arXiv preprint arXiv:1903.08542.

Kahn, Gregory, Villaflor, Adam, Ding, Bosen, Abbeel, Pieter, Levine, Sergey, 2018. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In: 2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018. pp. 1–8.

Kober, Jens, Peters, Jan, 2011. Policy search for motor primitives in robotics. Mach. Learn. 84 (1–2), 171–203.

Kohl, Nate, Stone, Peter, 2004. Policy Gradient reinforcement learning for fast quadrupedal locomotion. In: Proceedings of the IEEE International Conference on Robotics and Automation.

Koppejan, Rogier, Whiteson, Shimon, 2011. Neuroevolutionary reinforcement learning for generalized control of simulated helicopters. Evol. Intell. 4 (4), 219–241.

Kormushev, Petar, Calinon, Sylvain, G. Caldwell, Darwin, 2013. Reinforcement learning in robotics: Applications and real-world challenges. Robotics 2, 122–148.

Kulk, Jason, Welsh, James S., 2011. Evaluation of walk optimisation techniques for the nao robot. In: Humanoids. IEEE, pp. 306–311.

Lutz, Christian, Atmanspacher, Felix, Hornung, Armin, Bennewitz, Maren, 2012. Nao walking down a ramp autonomously. In: Video Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal.

Meriçli, Çetin, Veloso, Manuela, 2010. Biped walk learning through playback and corrective demonstration. In: AAAI 2010: Twenty-Fourth Conference on Artificial Intelligence.

Mescheder, Daniel, 2011. Learning to Walk A Self Optimizing Gait for the Nao (PhD thesis). Department of Knowledge Engineering, Maastrich University, Netherlands.

Miikkulainen, Risto, 2017. Evolution of neural networks. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. ACM, pp. 450–470.

Mouret, Jean-Baptiste, Chatzilygeroudis, Konstantinos I., 2017. 20 years of reality gap: a few thoughts about simulators in evolutionary robotics. In Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings, pp. 1121–1124.

Nikbin, Behzad, Ranjbar, Mohammad Reza, Sarjaz, Behrooz Shafiee, Shah-Hosseini, Hamed, 2011. Biped robot walking using particle swarm optimization. In: 16th Online World Conference on Soft Computing in Industrial Applications (WSC16).

Oßwald, Stefan, Görög, Attila, Hornung, Armin, Bennewitz, Maren, 2011. Autonomous climbing of spiral staircases with humanoids. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4844–4849.

Röfer, Thomas, Laue, Tim, 2014. On B-Human's code releases in the standard platform league – software architecture and impact. In: Behnke, Sven, Veloso, Manuela, Visser, Arnoud, Xiong, Rong (Eds.), RoboCup 2013: Robot World Cup XVII. In: Lecture Notes in Artificial Intelligence, 8371, Springer, pp. 648–655.

Shafii, Nima, Reis, Luí s, Lau, Nuno, 2010. Biped walking using coronal and sagittal movements based on truncated fourier series. Lecture Notes in Comput. Sci. 6556, 324–335.

Sutton, Richard S., Barto, Andrew G., 1998. Reinforcement Learning: An Introduction. The MIT Press.

Tijsma, Arryon D., Drugan, Madalina M., Wiering, Marco A., 2016. Comparing exploration strategies for q-learning in random stochastic mazes. In: 2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016, Athens, Greece, December 6-9, 2016. pp. 1–8.

Torres, Ernesto, Garrido, Leonardo, 2011. Automated generation of cpg-based locomotion for robot nao. In: Röfer, Thomas, Mayer, Norbert Michael, Savage, Jesus, Saranli, Uluc (Eds.), RoboCup. In: Lecture Notes in Computer Science, vol. 7416, Springer, pp. 461–471.

Webots, 2013. Commercial mobile robot simulation software.

Wu, Xiaoguang, Liu, Shaowei, Zhang, Tianci, Yang, Lei, Li, Yanhui, Wang, Tingjin, 2018. Motion control for biped robot via ddpg-based deep reinforcement learning. In: 2018 WRC Symposium on Advanced Robotics and Automation. pp. 40–45.

Yuan, Y., Li, Z., Zhao, T., Gan, D., 2019. Dmp-based motion generation for a walking exoskeleton robot using reinforcement learning. IEEE Trans. Ind. Electron. 1.