

Policy gradient learning for quadruped soccer robots

A. Cherubini^{a,b}, F. Giannone^a, L. Iocchi^{a,*}, D. Nardi^a, P.F. Palamara^a

^a Dipartimento di Informatica e Sistemistica, Sapienza University of Roma, Via Ariosto 25, 00185 Roma, Italy

^b INRIA/IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France

ARTICLE INFO

Article history:

Available online 2 April 2010

Keywords:

Reinforcement learning
Soccer robots

ABSTRACT

In real-world robotic applications, many factors, both at low level (e.g., vision, motion control and behaviors) and at high level (e.g., plans and strategies) determine the quality of the robot performance. Consequently, fine tuning of the parameters, in the implementation of the basic functionalities, as well as in the strategic decisions, is a key issue in robot software development. In recent years, machine learning techniques have been successfully used to find optimal parameters for typical robotic functionalities. However, one major drawback of learning techniques is time consumption: in practical applications, methods designed for physical robots must be effective with small amounts of data. In this paper, we present a method for concurrent learning of best strategy and optimal parameters using policy gradient reinforcement learning algorithm. The results of our experimental work in a simulated environment and on a real robot show a very high convergence rate.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

In order for robots to be useful in real-world applications, they must adapt to novel and changing environments. In most domains, the robot should respond to changes in its surroundings by adapting both its low-level skills (e.g., vision, motion control and basic behaviors) and the higher-level skills (e.g., plans or strategies as combination of basic behaviors), which depend on them.

In recent years, machine learning techniques have been used in several robotic applications, both for finding optimal parameters of specific behaviors and for determining the best combination of actions required to accomplish a complex task. In fact, machine learning approaches generate solutions with little human interaction, and explore the search space of possible solutions in a systematic way, whereas humans are often biased towards a small part of the space.

Despite a significant body of work on the subject, several issues need to be addressed, mainly due to the difficulties associated with using machine learning in the real world. Compared to other scenarios, learning on physical robots presents several additional challenges: it must be effective with small amounts of data and must converge in short time [1].

Following up on all these considerations, in this article we describe a reinforcement learning algorithm for learning at the same

time the best strategy (i.e., the best composition of basic behaviors) and the best parameters for the behaviors in the strategy. The proposed algorithm, named **PG-RC**, is based on the Policy Gradient (PG) learning technique for mobile robots, first introduced in [2], and exploits additional information on the system properties: *relevance* of parameters, and *contiguities* between strategies.

The contribution of this paper is twofold. Firstly, parameter and behavior learning are concurrently handled by the same algorithm. Secondly, the PG-RC algorithm guarantees faster training than the PG algorithm. The method has been tested in the application example of an attacker soccer robot in the RoboCup Standard Platform League (four-legged division).

2. Related work

Robot learning is a growing area of research at the intersection of robotics and machine learning. We can distinguish between different levels of learning: low level for sensing and control issues, and high level, for cognitive and behavior issues. We include in the first class – *parameter learning* – all problems where learning is aimed at fine tuning of the parameters used by the low-level algorithms, and in the second class – *behavior learning* – problems where learning aims at finding the optimal strategy, i.e. composition of behaviors, for accomplishing a given task.

In the first class of problems, one of the primary application areas is robot motion control, in all cases where the complete mathematical model is not known, and traditional optimization methods cannot be used. Gait optimization for legged robots is an example of this kind [1]. Parameter learning has proved very effective for improving other motion control tasks, such as robot

* Corresponding author.

E-mail addresses: Andrea.Cherubini@irisia.fr (A. Cherubini), giannone@algorithmica.it (F. Giannone), locchi@dis.uniroma1.it (L. Iocchi), Nardi@dis.uniroma1.it (D. Nardi), pier@cs.columbia.edu (P.F. Palamara).

grasping. In [3], this is achieved by applying the *layered learning* paradigm [4]: grasping parameters rely on previously learned walk parameters.

As for behavior learning, researchers have proved its utility in improving high-level robot tasks, e.g., navigation, exploration, path-planning. In the *behavioral decomposition* approach, the desired behavior is broken down into a set of simpler behaviors, which are activated or suppressed through a coordination mechanism. The structure of the simpler behavioral modules can emerge from a learning process, instead of being pre-designed. Nolfi [5] showed how an effective evolutionary approach can produce simpler and more robust solutions, than those designed by hand. The robot can evolve by adapting its behaviors to various constraints on the environment, on the desired task, and on the user satisfaction.

Typically, the aforementioned classes of learning (parameter and behavior learning) are not combined together in a single framework. One reason is the long time required to solve the optimization problem in the large search space including both parameters and behaviors. On the other hand, solving the two problems separately may lead to sub-optimal solutions. In this article, we present a fast parameter and behavior learning method based on policy gradient reinforcement learning [6], useful whenever behavior and parameter learning should be integrated for optimal task execution. Our method is based on the policy gradient technique and takes into account behavior similarities and parameter relevance during learning, to speed up convergence and handle parameter and behavior learning at the same time. Experiments have been conducted with two purposes: (1) verify the effectiveness of PG for learning complex robotic tasks; (2) evaluate the improvement of performance of PG when considering behavior similarities and parameter relevance (PG-RC).

3. Problem definition

In this paper, we consider learning a complex task composed of different behaviors. More specifically, we consider situations where a task \mathcal{T} can be accomplished by applying different strategies, and each strategy is a composition of different behaviors.

The learning problem we focus on is the following. A set $\mathcal{S} = \{S_1, \dots, S_{N_{str}}\}$ of N_{str} different *strategies* for accomplishing a certain task \mathcal{T} is given. Each strategy is implemented through a combination of behaviors: $\mathcal{B} = \{B_1, \dots, B_m\}$, each one characterized by a set of parameters $\Theta_{B_i} = \{\theta_1, \dots, \theta_{k_{B_i}}\}$. Let $\mathbf{X} \in \mathbf{R}^n$ be a representation of the parameters used by all the behaviors in the strategy S_v . Although, in general, different strategies have different parameter spaces, we will assume that the parameters of the behaviors for all the strategies can be represented as a vector in \mathbf{R}^n . Furthermore, we will denote with F the fitness function used to evaluate strategies and parameters and thus with $F(\mathbf{X})$ the evaluation of the parameters \mathbf{X} within the strategy S_v . The objective of the learning process is thus to learn the best choice of the strategy S_{v^*} , along with the optimal parameters \mathbf{X}^{v^*} that ensure the best performance on the execution of the task, i.e., that maximize $F(\mathbf{X})$.

The considered problem deals at the same time with: (1) the choice of an optimal strategy, (2) an optimization problem in the behavior parameters space. In general, the choice about which strategy must be selected to perform a task is not easy. This is firstly due to the fact that in complex scenarios, the winning strategy depends on the context (i.e., the current situation of the environment around the robot). Secondly, some strategies can perform similarly in a given context: this similarity (*contiguity*) can be used to speed up the learning process. Thirdly, a coarse initial design of the learning algorithm can lead to overestimation of the search space dimensions. In particular, this occurs when parameters which have

little *relevance* on the performance are learned. Based on these considerations, we will show that the speed of a robot task learning algorithm can be improved by: (1) detecting and exploiting similarities between learned strategies, and (2) updating the search space dimensions, by removing irrelevant parameters during learning.

4. Complex task learning with policy gradient

A solution to the problem defined in the previous section can be obtained by applying a Policy Gradient learning task for each strategy and then selecting the pair $(S_{v^*}, \mathbf{X}^{v^*})$ that returns the highest fitness value for task \mathcal{T} . This procedure requires N_{str} executions of the PG algorithm (one execution for each strategy). The algorithm presented in this article is an approximation of the above solution, that guarantees a higher convergence rate when a small number of experiments is available.

The PG algorithm estimates the gradient of the objective function in the parameter space and follows it towards a local optimum. The parameter optimization approach starts from an initial parameter set \mathbf{X}^0 and proceeds to estimate the partial derivative of $F(\mathbf{X}^0)$ with respect to each parameter. From the initial set \mathbf{X}^0 , p randomly generated *policies* $m\mathbf{X}$ ($m = 1, \dots, p$), near \mathbf{X}^0 , are evaluated. The number of policies p is proportional to the search space dimension: $p = \lambda k$. Each of the p policies is generated as: $m\mathbf{X} = \mathbf{X}^0 + [\rho_1, \dots, \rho_k]^T$, and each ρ_j is chosen randomly in the set $\{-\epsilon_j, 0, +\epsilon_j\}$. After evaluating the objective function at each $m\mathbf{X}$, average scores $\bar{F}_{-\epsilon_j}$, $\bar{F}_{0,j}$, and $\bar{F}_{+\epsilon_j}$ are computed and used to construct an estimate of the *gradient*, $\nabla \mathbf{X}^0$, which is then normalized, multiplied by a scalar constant *step-size* η and added to \mathbf{X}^0 , to begin the next iteration i ($i = 1, \dots, N_{iter}$).

Although effective, the method described above does not exploit two important factors: *relevance* of parameters within each strategy and *contiguities* between strategies. The algorithm presented in this article is called PG-RC and improves learning performance by exploiting *Relevance* and *Contiguity*. Let us define, at every iteration step $i \neq 0$, the following metrics:

- *Relevance* of parameter j for a given strategy v : it is the norm of the weighted average of the j th gradient component:

$$R^i(v, j) = \frac{\sum_{i=1}^{\bar{i}} \lambda^{\bar{i}-i} \nabla^v X_j^i}{\sum_{i=1}^{\bar{i}} \lambda^{\bar{i}-i}}$$

where $0 < \lambda < 1$ is a *forgetting factor* that reduces the weight of old data. Small values of the *relevance* imply that the estimated gradient varies ‘slightly’ along the j th component during the learning process: hence, we assume that the corresponding parameter has little relevance on the system performance. We denote with ${}^v J^i \subseteq \{1, \dots, k\}$ the subset of *relevant* parameters (i.e., parameters j with ‘high’ $R^i(v, j)$) for learning strategy v at iteration i . ${}^v J^i$ is updated at each iteration step i during the learning process, depending on the values of $R^i(v, j)$.

- *Contiguity* between strategies S_v and S_w :

$$C^i(v, w) = \frac{1}{d(v\mathbf{X}^i, w\mathbf{X}^i) + 1} \quad \text{with}$$

$$d(v\mathbf{X}^i, w\mathbf{X}^i) = \sqrt{\sum_{j=1}^k (vX_j^i - wX_j^i)^2}$$

where $d(\cdot, \cdot)$ is the Euclidean distance in the parameter space. We also denote with ${}^v P^i$ the set (*cluster*) containing all points \mathbf{X}^i which are ‘near’ \mathbf{X}^i .

PG-RC ALGORITHM

```

INPUT:  $\mathcal{S}$ ,  $v \underline{X}^0$ ,  $T_r$ ,  $T_c$ ,  $N_{iter}$ 
OUTPUT:  $S_{v^*}$ ,  $v^* \underline{X}^*$ 

1 BEGIN
2 initialize for all  $v$ :  $v \underline{X} \leftarrow v \underline{X}^0$ ,  $v J^0 \equiv \{1, \dots, v k^0\}$ 
3 for each iteration  $i = 1$  to  $N_{iter}$ 
4   for each strategy  $v = 1$  to  $N_{str}$ 
5     // apply PG to the relevant parameters
6     generate  $v \underline{X}^i$  perturbations of  $v \underline{X}^{i-1}$  on  $v J^{i-1}$ 
7     evaluate  $F(v \underline{X}^i)$  at all  $v p^{i-1}$  policies  $v \underline{X}^i$  for task  $\mathcal{T}$ 
8      $v J^i \leftarrow \emptyset$ 
9     for each parameter  $j \in v J^{i-1}$ 
10      evaluate  $\bar{F}_{-\epsilon, j}$ ,  $\bar{F}_{0, j}$ , and  $\bar{F}_{+\epsilon, j}$ 
11      if  $\bar{F}_{0, j} > \bar{F}_{-\epsilon, j}$  and  $\bar{F}_{0, j} > \bar{F}_{+\epsilon, j}$ 
12         $\nabla^v \underline{X}_j^i \leftarrow 0$ 
13      else
14         $\nabla^v \underline{X}_j^i \leftarrow \bar{F}_{+\epsilon, j} - \bar{F}_{-\epsilon, j}$ 
15      endif
16      evaluate  $R^i(v, j)$ 
17      if  $R^i(v, j) > T_r$ 
18         $v J^i \leftarrow v J^i \cup j$ 
19      endif
20    endfor // parameters
21     $\nabla^v \underline{X}^i \leftarrow \eta \times \frac{\nabla^v \underline{X}^i}{|\nabla^v \underline{X}^i|}$ 
22     $v \underline{X}^i \leftarrow v \underline{X}^{i-1} + \nabla^v \underline{X}^i$ 
23  endfor // strategies
24  // evaluate contiguities of each pair of strategies
25  for each strategy  $v = 1$  to  $N_{str}$ 
26    for each strategy  $w = 1$  to  $N_{str}$ 
27      evaluate  $C^i(v, w)$ 
28      if  $C^i(v, w) > T_c$ 
29         $v P^i \leftarrow v P^i \cup w \underline{X}^i$ 
30      endif
31    endfor // strategies  $w$ 
32    evaluate  $F(\underline{X})$  at all  $\underline{X} \in v P^i$ 
33     $v \underline{X}^i \leftarrow \arg \max_{\underline{X} \in v P^i} F(\underline{X})$ 
34  endfor // strategies  $v$ 
35 endfor // iterations
36 // return best values
37 for each strategy  $v = 1$  to  $N_{str}$ 
38    $v \underline{X}^* \leftarrow \arg \max_i F(v \underline{X}^i)$ 
39 endfor // strategies  $v$ 
40  $v^* \leftarrow \arg \max F(v \underline{X}^*)$ 
41 return  $(S_{v^*}, v^* \underline{X}^*)$ 
42 END

```

Fig. 1. Pseudocode for the PG-RC algorithm.

The PG-RC algorithm is shown in Fig. 1. The differences with the PG algorithm are: (1) for each strategy, the subset $v J^i$ is updated (lines 16–18), and at each iteration, policies are generated by perturbing only *relevant* parameters (line 9). (2) for *contiguous* strategies, points reached at each learning iteration and belonging to $v P^i$ are evaluated, and the best point is chosen as initial point for the next iteration for each of the contiguous strategies (lines 32–33).

The PG algorithm is proved to converge to a local optimum \underline{X}^* if a sufficient number of iterations is performed. The total number of experiments necessary to complete N_{iter} iterations in PG is $N_{PG} = p N_{iter} = \Lambda k N_{iter}$. The PG-RC algorithm, instead, progressively reduces the number of relevant parameters (i.e., $|J^i|$) and thus the number of policies $p^i = \Lambda |J^i|$ also diminishes. The total number of experiments necessary to complete N_{iter} in PG-RC is $N_{PGPR} = \sum_{i=0}^{N_{iter}} p^i = \Lambda \sum_{i=0}^{N_{iter}} |J^i|$. Hence, in general, $N_{PGPR} < N_{PG}$.

The second modification (i.e., the use of contiguity) is based on the assumption that *similar* strategies have *similar* optima. In fact, at iterations where two (or more) strategy optimization problems have given *similar solutions* (i.e., solutions with high contiguity),

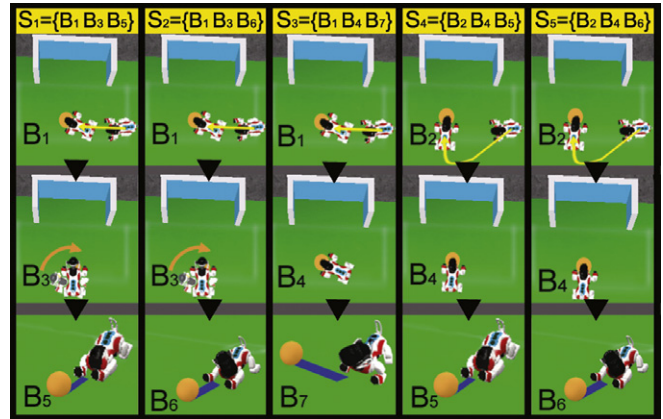


Fig. 2. Five strategies that can be used in the attacking task.

such solutions are tested on both strategies. This requires 2 extra experiments each time the condition at line 28 is verified, but it provides a way of fusing the N_{str} optimization problems, instead of considering them separately. The advantage of this modification will be shown empirically in the following section.

In summary, the PG-RC algorithm unifies parameter and behavior optimization problems, by taking into account parameter relevance (which contributes mainly to the first problem), and strategy contiguity (for the latter problem).

5. Experiments with a soccer robot

Concurrent behavior and parameter learning is useful in many complex high dimensional systems. Here, we present an example of application of the proposed method: a robot playing soccer within the RoboCup Standard Platform League competitions. One of the main tasks to be accomplished in this scenario is to approach the ball and kick it to the opponent goal. Many strategies can be defined to accomplish this task, but a winning strategy is difficult to identify since it depends on many factors: position of the robot and of the ball in the field, position of the other robots, etc. Besides, each behavior has several parameters to be tuned: walking gait parameters, kick parameters, etc. Learning such a complex task requires to define a strategy (as a combination of behaviors) and tune the parameters of the behaviors involved in the strategy.

The major contribution of the experiments is the comparison between the policy gradient methods presented in Section 4. In practice, we focus on the following task \mathcal{T} : *ball 50 cm in front of robot, and direction from ball to goal orthogonal to the robot sagittal axis, with the goal on the left of the robot*. The objective of the learning algorithm is to find the most suitable strategy S_{v^*} and the corresponding optimal parameters $v^* \underline{X}^*$ for a given task.

Behaviors.

For learning the *attacking* task, a set of seven behaviors $\mathcal{B}_P = \{B_1, \dots, B_7\}$ (see Fig. 2) has been considered. These behaviors (see [7] for further details) can be classified in three subsets:

- **Ball approaching behaviors:**
 - B_1 : *fast ball approach* – the path length is minimized by an omnidirectional walk;
 - B_2 : *aligning ball approach* – while the robot approaches the ball, its heading is concurrently oriented towards the opponent goal;
- **Ball controlling behaviors:**
 - B_3 : *rotational ball carrying* – the robot grasps the ball with its chin, and subsequently turns its heading towards the opponent goal;

- B_4 : *gradual ball reaching* – the robot gradually decreases its translational and rotational velocities in order to reach the ball without knocking it;
- **Ball kicking behaviors:**
 - B_5 : *head straight kick* – the robot ‘dives’ on the ball and hits it with the head – the head pan angle is kept null during this kick;
 - B_6 : *head spanning kick* – the robot ‘dives’ on the ball and hits it with varying head pan: the kick is more powerful but less precise than B_5 ;
 - B_7 : *lateral head kick* – the robot hits the ball laterally with a varying head pan angle.

Parameters.

Each behavior B is characterized by a set of parameters Θ_B . The parameters that we have chosen spring from the long lasting experience of our Robocup team, SPQRLegged¹ and are commonly used to control the robot attack.

The parameters are box-constrained (i.e., defined within a fixed interval of values) due to the physical characteristics of the system, and we call Δ_j the interval of values for each θ_j .

Speed and stability of the ball approach are mainly characterized by the *walking gait parameters*, which define the kinematic characteristics of the walk. The trot gait is used, with a half-ellipsoidal locus shape for foot motion, and contact between the base of the locus and the ground generates body motion. The gait is characterized by the eleven walking gait parameters: $\Theta_{WG} = \{\theta_{WG,1}, \dots, \theta_{WG,11}\}$.

Walking gait parameters have little influence on the quality of ball control, which relies mainly on the way the robot slows down and stops near the ball (i.e. on the *ball controlling parameters*). We use translational and rotational slowing down velocities that are respectively proportional to the ball distance and to the relative robot-ball angle. Thus, ball control is influenced by the four parameters: $\Theta_{BC} = \{\theta_{BC,1}, \dots, \theta_{BC,4}\}$.

The robot kicks are generated by a sequence of fixed joint positions (for AIBO’s 15 joints), at three stages of the kick, related to a timing information, stating for how long the joint values must be kept (i.e. the kick ‘speed’). Motion of the back leg joints is fixed (since it has little influence on the kick quality), whereas motion of the head and front legs should be learned. In practice, twenty three *ball kicking parameters* $\Theta_{BK} = \{\theta_{BK,1}, \dots, \theta_{BK,23}\}$ are used.

Strategies.

A strategy is a combination of behaviors. In this paper we consider only a simple form of behavior composition: chaining between a ball approaching, a ball controlling, and a ball kicking behavior. Thus we consider *strategies* as sequences of three behaviors. For example, $\{B_1; B_3; B_5\}$ is a possible strategy for the attacking task. In the experiments reported below we considered 5 different strategies as different combinations of the above described behaviors:

$$S_1 = \{B_1 B_3 B_5\} \quad S_2 = \{B_1 B_3 B_6\} \quad S_3 = \{B_1 B_4 B_7\} \\ S_4 = \{B_2 B_4 B_5\} \quad S_5 = \{B_2 B_4 B_6\}.$$

Learning.

The learning process consists in choosing the best strategy and optimizing the parameter sets: Θ_{WG} , Θ_{BC} , and Θ_{BK} . These sets are

represented as a vector:

$$\underline{X} = [\theta_{WG,1} \dots \theta_{WG,11} \theta_{BC,1} \dots \theta_{BC,4} \theta_{BK,1} \dots \theta_{BK,23}]^T \in \mathbf{R}^{38} \quad (1)$$

that also represents a candidate solution of the optimization problem. As already mentioned, these sets might include parameters which are irrelevant for a certain strategy, and the learning algorithm will deal with them.

The very large dimensions of the search space, and the system characteristics, suggest the use of the *layered learning* paradigm [4] for optimizing this problem. In fact, given a hierarchical task decomposition, layered learning allows for learning at each level of the hierarchy, with learning at each level directly affecting learning at the next higher level. The *incremental learning* approach that we use is inspired by the layered learning paradigm; however, in contrast with typical layered learning, we utilize the same learning method for each layer of the hierarchy. Specifically, we decompose optimization of the strategy parameters in the following three optimization subtasks (layers):

- L_1 : find $\underline{X}_{1,opt} = [\theta_{WG,1} \dots \theta_{WG,11}]_{opt}^T \in \mathbf{R}^{11}$ for ‘best’ walk;
- L_2 : find $\underline{X}_{2,opt} = [\theta_{BC,1} \dots \theta_{BC,4}]_{opt}^T \in \mathbf{R}^4$ for ‘best’ ball control, given $\underline{X}_{1,opt}$ found by L_1 ;
- L_3 : find $\underline{X}_{3,opt} = [\theta_{BK,1} \dots \theta_{BK,23}]_{opt}^T \in \mathbf{R}^{23}$ for ‘best’ kicking, given $\underline{X}_{1,opt}$ and $\underline{X}_{2,opt}$ found by L_1 and L_2 .

In fact, reinforcement learning is not suitable for dealing with the 38 dimensional vector in Eq. (1). However, by utilizing the incremental learning approach, to decompose the optimization problem in smaller subproblems, our algorithm proves effective, as shown later. The solution of the attacker strategy optimization problem can then be obtained as:

$$\underline{X}_{opt} = [\underline{X}_{1,opt} \quad \underline{X}_{2,opt} \quad \underline{X}_{3,opt}]^T.$$

The appropriate choice of the objective function for optimization is fundamental. The function for evaluating the quality of an attacking strategy – the *fitness function* – must take into account: the quality (speed and precision) of the robot motion for approaching the ball, the quality of ball control, and the quality (power and precision) of the kick. Hence, we adopt the following objective function for evaluating all the strategies:

$$F(\underline{X}) = K_{WG} f_{WG}(\underline{X}) + K_{BC} f_{BC}(\underline{X}) + K_{BK} f_{BK}(\underline{X}) \quad (2)$$

where f_{WG} , f_{BC} , f_{BK} indicate the quality of the ball approaching, ball controlling, and ball kicking behavior respectively, and K_{WG} , K_{BC} , and K_{BK} are positive weights indicating the significance desired for each of the three aspects in the learning process.

Experiments with a simulated robot.

Learning task \mathcal{T} is initially implemented within the 3D AIBO simulator [8] embedded in USARSim² before experimentation on the real robot. In fact, in [7], we have shown that the learning configurations tuned in the simulated environment can be successfully used for learning on the real robot. In the simulations, the fitness components f_{WG} , f_{BC} , f_{BK} in (2) are calculated from the robot and ball positions, which are perfectly known in USARSim.

We utilize an incremental learning approach using the policy gradient algorithms (PG and PG-RC) described before. For both algorithms, and for all 3 layers, we choose the policy gradient configuration: $\epsilon_j = 0.1 \Delta_j$, $\eta = 3$, and $\Lambda = 0.7$. For PG-RC, we set: $T_r = 0.03$ and $\lambda = 0.8$, and $T_c = 0.002$. Obviously, the evolution of the optimization process is related to the choice of the starting point, i.e., of the initial parameter set. For this reason,

¹ spqr.dis.uniroma1.it.

² usarsim.sourceforge.net.

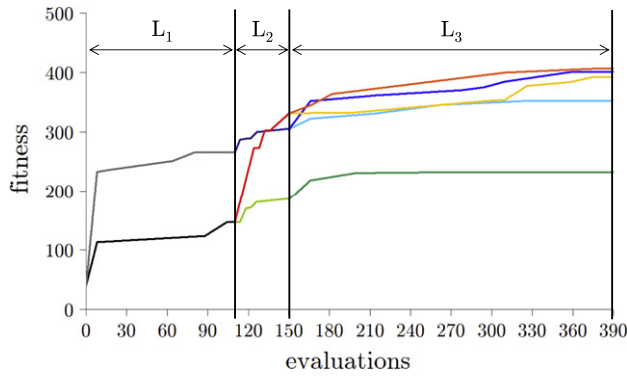


Fig. 3. Fitnesses obtained during USARSim training with PG algorithm for strategies at layers: L_1 (strategies S_1, S_2 and S_3 in black, S_4 and S_5 in gray), L_2 (strategies S_1, S_2 in solid black (red in web version), S_3 in dashed black (green in web version), S_4 and S_5 in gray (blue in web version)) and L_3 : S_1 (solid black (orange in web version)), S_2 (dashed black (yellow in web version)), S_3 (dotted black (green in web version)), S_4 (solid gray (blue in web version)) and S_5 (dashed gray (cyan in web version)).

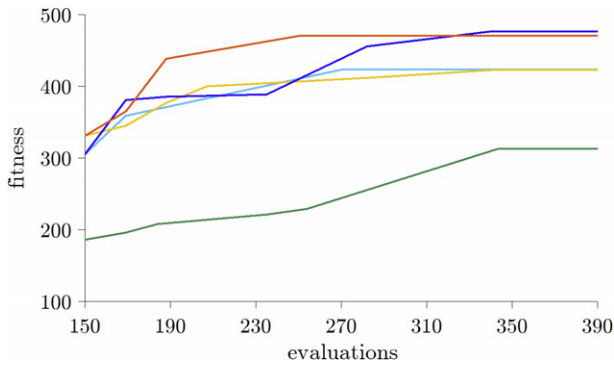


Fig. 4. Fitnesses obtained during USARSim training at layer L_3 with PG-RC algorithm for strategies: S_1 (solid black (orange in web version)), S_2 (dashed black (yellow in web version)), S_3 (dotted black (green in web version)), S_4 (solid gray (blue in web version)) and S_5 (dashed gray (cyan in web version)).

we have decided to use the same initial parameter set 0X for starting learning on each strategy. The set 0X was manually tuned in order to obtain near fitnesses for all five strategies, and therefore facilitate the comparison. Since there is significant noise in the experiments, each set of parameters is evaluated twice, and the resulting fitness *evaluated* for that set, is computed by averaging over the two experiments. In USARSim, since ‘hardware consumption’ is not an issue, for each layer, we compare the learning algorithms after $10 k_i$ evaluations have been performed (e.g. for L_3 , 240 evaluations, i.e., 480 experiments).

Two series of experiments have been performed. The first experiments consist in the application of incremental learning using PG algorithm: for the three layers L_1, L_2 and L_3 described before, we have obtained three optimal solutions of the relative sub-tasks: ${}^vX_{1,opt,PG}, {}^vX_{2,opt,PG}, {}^vX_{3,opt,PG}$, for each strategy v . The



Fig. 5. Evolution of the simulated robot learning using PG-RC at layer L_3 for strategy S_3 . Left to Right: evaluations 150, 168, 185, 202.

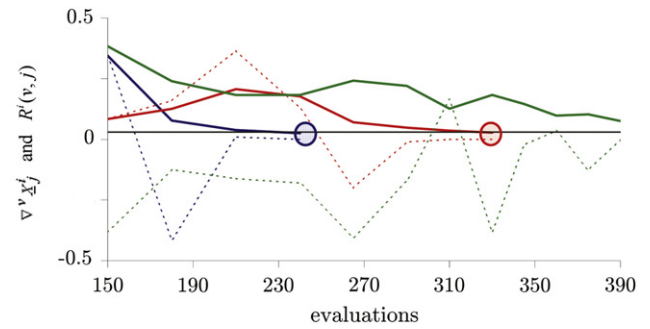


Fig. 6. Evolution of *gradient component* (gray (dashed line in web version)) and *relevance* (black (continuous line in web version)) of: $\theta_{BK,13}$ (solid (green in web version)), $\theta_{BK,13}$ (dashed (red in web version)) and $\theta_{BK,24}$ (dotted (blue in web version)) during USARSim training of S_2 at layer L_3 : circles indicate the evaluation step where the parameter becomes irrelevant (the straight black line indicates T_r).

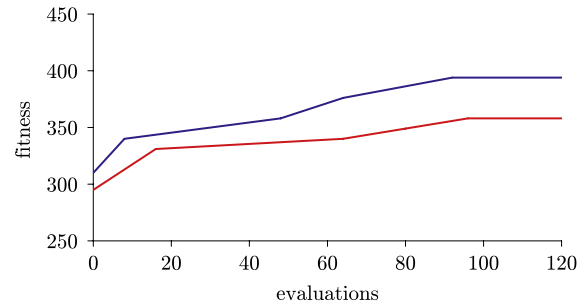


Fig. 7. Fitnesses during AIBO training with start point derived from USARSim training: PG in gray (red in web version), PG-RC in black (blue in web version).

second experiments consist in the comparison of PG and PG-RC learning algorithms. This has been performed only at layer L_3 , thus computing ${}^vX_{3,opt,PG-RC}$, for each strategy v . In this second series of experiments, we emphasized the capability of PG-RC to filter out irrelevant parameters, by adding a 24th *irrelevant* parameter $\theta_{BK,24}$ (which is not used by the robot for kicking the ball) to the set Θ_{BK} .

We consider a learning process of 390 evaluations. The results of learning L_1, L_2 and L_3 for the 5 strategies with PG algorithm are shown in Fig. 3. Notice that, thanks to the incremental approach, at each layer, strategies sharing common behaviors are optimized simultaneously. Thus only two different sessions are needed at layer L_1 and three at layer L_2 . The figure also shows that at the end of the optimization process, similar fitnesses are obtained with strategies S_1 and S_4 , which outperform the 3 other strategies.

The results obtained with the PG-RC algorithm on learning layer L_3 for the 5 strategies are instead presented in Fig. 4. In this case, the graphs start from evaluation 150, since the previous evaluations are relative to layers L_1 and L_2 . For all the strategies, the final fitness obtained with PG-RC outperforms the fitnesses obtained with the PG algorithm. The best improvements are for strategies S_3 (the fitness increases by 35% with respect to PG) and S_5 (increase of 21%). Strategy S_4 achieves the best performance at the

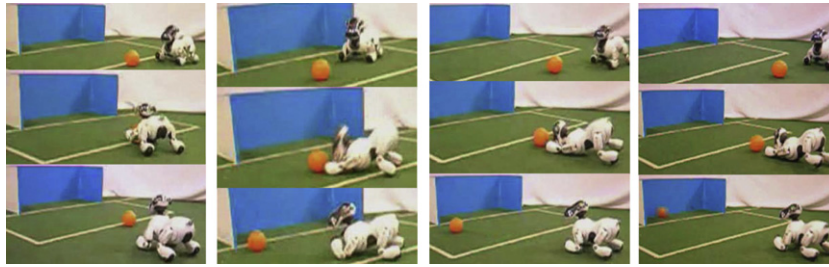


Fig. 8. Evolution of the real AIBO robot learning at layer L_3 for strategy S_4 . Left to Right: evaluations 1, 48, 66, 95.

end of PG–RC learning. Snapshots of the simulations are shown in Fig. 5 as the robots learn layer L_3 for strategy S_3 with PG–RC. The figure shows, from left to right, four evaluations during learning. The snapshots show how the kick quality progresses during learning: as the process evolves, the movement improves, fostering the hitting effectiveness, and consequently increasing both the accuracy and strength of the kick.

During the learning process, the algorithm computes parameter relevance and fixes the parameters that do not contribute to improve the solution. At the end of the learning process, for all 5 strategies, approximately 50% of the parameters have been fixed (e.g., 11 parameters for S_1 , 13 parameters for S_4). Fig. 6 plots the gradient component and relevance of parameters $\theta_{BK,3}$, $\theta_{BK,13}$ and $\theta_{BK,24}$ during a learning process. Note that, as expected, $\theta_{BK,24}$ is filtered out very early in the learning process. More specifically, most of the experiments reveal the irrelevance of the leg positions at an intermediate stage of the kick. This implies that it could be sufficient to learn the leg positions at the first and final stage of the kick. On the other hand, examples of relevant parameters selected by the algorithm, are: the head tilt positions (for all strategies) (e.g., $\theta_{BK,3}$ is the initial head tilt value), the front hip positions for S_1 , S_2 , S_4 and S_5 (which use kicks B_5 and B_6) and the front knee positions for S_3 (kick B_7).

Moreover, the algorithm also takes advantage of the contiguities between strategies S_1 , S_2 and S_3 and also between strategies S_1 , S_4 and S_5 , to speed-up convergence to a good solution.

Experiments with the real robot.

Finally, both algorithms are tested on the real robot starting from the results of simulator optimization. We performed experiments on learning layer L_3 for the best strategy obtained in simulation, i.e., $v^* = 4$, with both PG and PG–RC algorithms. The fitness components f_{WG} , f_{BC} , f_{BK} in (2) are calculated by combining the measures of the robot and ball displacements, with a qualitative assessment from a human supervisor. Moreover, each set of parameters is evaluated twice, and the resulting fitness is obtained by averaging the results of each test. We consider a learning process of 120 evaluations. The results are shown in Fig. 7 and snapshots of the experiments are shown in Fig. 8.³

Again, we observe that PG–RC outperforms PG. At the end of the experiments, 13 of the initial 24 parameters are preserved. As in the simulations, irrelevant parameter $\theta_{BK,24}$ is filtered out at the beginning of the learning process. On the other hand, the front hip leg positions are relevant throughout the experiments. Their position is fundamental for the kick (specifically, in strategy S_4 , kick B_5), to avoid that the head collides with the legs before the ball is hit.

6. Conclusions

In this paper, we presented a method for concurrent learning of best strategy and optimal parameters, by extending the policy

gradient reinforcement learning algorithm. The proposed method guarantees fast convergence by exploiting information on the system properties, while learning. In particular, the contiguities between strategies, and the parameter relevance, are estimated and utilized by the algorithm during training. Parameter relevance enables reduction of the search space size during learning (and consequent increase in convergence speed), while contiguities improve optimization by search for maxima among similar strategies.

The proposed learning technique has been applied to the soccer attacking task. It has been implemented in USARSim and extensively experimented in laboratory on AIBO, showing a notable improvement in the learned robot performance with respect to robot performance learned with classic policy gradient. Our technique outperforms classic policy gradient when the comparison is carried out over the same number of evaluations. Moreover, the convergence rate of our technique is higher than that of classic policy gradient: the same fitness value (e.g., in USARSim, 400) is attained much earlier (145 evaluations in advance). Fast convergence of learning is a key feature in robot applications, where hardware consumption is a major issue.

Development of a robot task learning algorithm that identifies the system major characteristics during training could be the object of further work. For example, learning adaptive strategies that depend on the situation the robot is facing is an interesting challenge for accomplishment of complex tasks in dynamic environments.

References

- [1] S.K. Chalup, C.L. Murch, M.J. Quinlan, Machine learning with AIBO robots in the four-legged league of RoboCup, IEEE Transactions on Systems, Man and Cybernetics, Part C (2006).
- [2] N. Kohl, P. Stone, Policy gradient reinforcement learning for fast quadrupedal locomotion, in: Proc. of IEEE International Conference on Robotics and Automation, 2004.
- [3] P. Fiedelman, P. Stone, The chin pinch: a case study in skill learning on a legged robot, in: Proc. of 10th International Robocup Symposium, 2006.
- [4] P. Stone, M. Veloso, Layered learning, in: Proc. of 11th European Conference on Machine Learning, 2000.
- [5] S. Nolfi, Evolutionary robotics: Exploiting the full power of self-organization, Connection Science 10 (1998) 167–183.
- [6] R. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, Advances in Neural Information Processing Systems 12 (2000) 1057–1063.
- [7] A. Cherubini, F. Giannone, L. Iocchi, Layered learning for a soccer legged robot helped with a 3D simulator, in: RoboCup 2007: Robot Soccer World Cup XI, 2008, pp. 385–392.
- [8] M. Zaratti, M. Fratarcangeli, L. Iocchi, A 3D simulator of multiple legged robots based on USARSim, in: In RoboCup 2006: Robot Soccer World Cup X, 2007, pp. 13–24.



A. Cherubini received the M.Sc. degree (“Laurea”) in Mechanical Engineering in 2001 from the University of Rome “La Sapienza”, the M.Sc. degree in Control Systems in 2003 from the University of Sheffield, UK, and the Ph.D. degree in Systems Engineering in 2008 from the University of Rome “La Sapienza”.

During his Ph.D. programme (2004–2007), he was a visiting scientist at the Lagadic group at INRIA Rennes - Bretagne Atlantique in Rennes (France), where he is currently working as post-doctoral fellow.

His research interests include: visual servoing for mobile robotic applications, robot learning, nonholonomic robot navigation, assistive robotics and legged locomotion.

³ Videos available at: www.dis.uniroma1.it/~labrob/people/cherubini/assets/files/RAS09-extendedPGquadruped.mpeg.



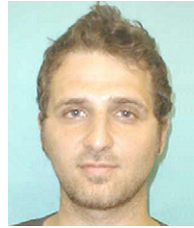
F. Giannone received the B.Sc. degree in Computer Engineering from Università di Roma “La Sapienza” in December 2005. Her main interests are machine learning and behavior modeling through Petri nets applied to Cognitive Robotics.



D. Nardi is Full Professor at Dipartimento di Informatica e Sistemistica, Sapienza University of Rome, Italy. His main research interests include various aspects of knowledge representation and reasoning, such as description logics and nonmonotonic reasoning, cognitive robotics, multi-agent and multi-robot systems.



L. Iocchi received his Master (Laurea) degree in 1995 and his Ph.D. in 1999 from Sapienza University of Rome. He is currently Assistant Professor at Department of Computer and Systems Science, Sapienza University of Rome, Italy. His main research interests are in the areas of cognitive robotics, action planning, multi-robot coordination, robot perception, robot learning, stereo vision, and vision based applications. He is author of more than 100 referred papers in international journals and conferences.



P.F. Palamara received his B.Sc. (2005) and M.Sc. (2008) in Computer Science from the University of Rome “Sapienza”, and an M.Sc. in Computer science from Columbia University (2009). He is currently a doctoral student in the Itsik Pe’er Lab of Computational Genetics, Columbia University. His main research interests are Computational Genetics, Machine Learning and Cognitive Robotics.