Layered Learning for a Soccer Legged Robot Helped with a 3D Simulator

A. Cherubini, F. Giannone, and L. Iocchi

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Ariosto 25, 00185 Roma, Italy
{cherubini,iocchi}@dis.uniroma1.it

Abstract. Mobile robots can benefit from machine learning approaches for improving their behaviors in performing complex activities. In recent years, these techniques have been used to find optimal parameter sets for many behaviors. In particular, layered learning has been proposed to improve learning rate in robot learning tasks. In this paper, we consider a layered learning approach for learning optimal parameters of basic control routines, behaviours and strategy selection. We compare three different methods in the different layers: genetic algorithm, Nelder-Mead, and policy gradient. Moreover, we study how to use a 3D simulator for speeding up robot learning. The results of our experimental work on AIBO robots are useful not only to state differences and similarities between different robot learning approaches used within the layered learning framework, but also to evaluate a more effective learning methodology that makes use of a simulator.

1 Introduction

In order for robots to be useful for many real-world applications, they must be able to effectively perform complex tasks. For this purpose, a popular research activity is the annual RoboCup soccer competition[1]. In this domain, the robot should be able to respond to changes in its surroundings by adapting both its low-level skills (e.g., the walking style) and the higher-level skills (e.g., the behaviour) which depend on them. Firstly, creating effective motion for walking and kicking the ball is a challenging task, since there are many parameters to be set, and since successful motions strongly depend on many factors, including: playing surface, robot hardware, and game situation. In recent years, machine learning techniques have been used to find optimal parameter sets. Secondly, in Robocup matches, the correct choice of the best behaviours required to accomplish a certain task (e.g., to score a goal) is fundamental for success. Machine learning techniques have been used in this field as well, in order to adapt the behaviours to the given game situation.

[1] Here, we focus on the RoboCup Four-Legged league (see www.tzi.de/4legged/).

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 385–392, 2008.
© Springer-Verlag Berlin Heidelberg 2008

Indeed, robot learning has been used both for fine-tuning of the parameters used by the low level algorithms – *parameter learning* – and for finding the optimal composition of simple behaviors for accomplishing a certain task – *behavior learning*. In the first class, an important application area is robot motion control, such as gait optimization for legged robots. For example, a Genetic Algorithm has been used for optimization of the vector of quadruped walk parameters in [3]. Kohl and Stone [6], empirically compared four machine learning algorithms for quadruped walk optimization. Parameter learning has proved very effective for improving other motion control tasks, such as grasping. This task is achieved in [5] by applying the *layered learning* paradigm [9]: grasping parameters rely on previously learned walk parameters.

On the other hand, researchers proved the utility of behavior learning for improving high level tasks, e.g. navigation [4], path-planning, and multi-robot cooperation [8]. To our knowledge, parameter and behaviour learning have rarely been joined in a single framework. In [1], this has been done by gradually developing cognitive capabilities, starting from abilities for detecting and recognizing objects, up to task execution.

Besides, experimental comparison of different learning methodologies has been rarely addressed. Here, we focus on learning a task for a AIBO soccer robot. Behavior learning and parameter learning are integrated, and we present an experimental evaluation of a layered learning approach [9] using three different learning techniques: genetic algorithm (GA), Nelder-Mead (NM), and policy gradient (PG). Finally, we study how to use a 3D simulator for speeding up robot learning. The results of our experimental work can be summarized as follows: 1) layered learning is very effective in the complex scenario considered in this paper; 2) using a 3D simulator can speed up the learning process on real robots; 3) the learned low-level parameters are strongly related to the desired behavior. We have successfully experimented the presented learning methodology in preparation of RoboCup 2006, showing a notable improvement of performance of the robots in our team.

2 Problem Definition

In this paper, we consider learning a complex task as a composition of different behaviors. More specifically, we consider situations in which a task $\mathcal{T}$ can be accomplished by applying different strategies, where each strategy is a composition of different behaviors. Each behavior B is characterized by a set of parameters $\Theta_B = \{\theta_1, \ldots, \theta_{k_B}\}$. Notice that a behavior can be present in different strategies and possibly requires the definition of different parameters depending on the situation in which it is used. The learning problem is thus twofold: on one hand, behavior learning is needed to select the best strategy, i.e., the best composition of basic behaviors; on the other hand, each behavior needs fine tuning of each parameter. In the next section we present a learning methodology that integrates behavior and parameter learning for a complex robot task.

To make this problem more clear we will present the application example in which we have tested our method. Consider a robot playing soccer within the RoboCup Four-Legged league competitions. One of the main tasks to be accomplished is to approach the ball and kick it to the opponent goal. This is a complex task that requires the integration of different behaviors in different ways (i.e., strategies). 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 such a strategy.

3 System Description

In this section, we describe learning the *attacking* task for a soccer robot in the Four-Legged League.

3.1 Behaviors and Parameters

For learning the *attacking* task, a set of six behaviors $\mathcal{B}_P = \{B_1, \ldots, B_6\}$ has been considered. These behaviors (see Figure 1) 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 oriented towards the goal);
- *ball carrying* behaviors (aimed at grasping the ball with the robot chin, and orienting the robot heading towards the opponent goal): B_3: *rotational ball carrying* (alignment is achieved by pure rotation), B_4: *rototranslational ball carrying* (alignment is achieved by a rototranslational movement);
- *ball kicking* behaviors (realized by direct kinematics control of the 15 joint positions): B_5: *head straight kick* (the robot 'dives' on the ball and hits it with the head), B_6: *head spanning kick* (the robot 'dives' on the ball and hits it with varying head pan).

A strategy is a combination of behaviors. Here, we consider only a simple form of behavior composition: chaining. Thus, we consider *strategies* as sequences of behaviors. For example, $\{B_1; B_4; B_5\}$ is a possible strategy for the attacking task. Each behavior B is characterized by a set of parameters Θ_B. Speed and

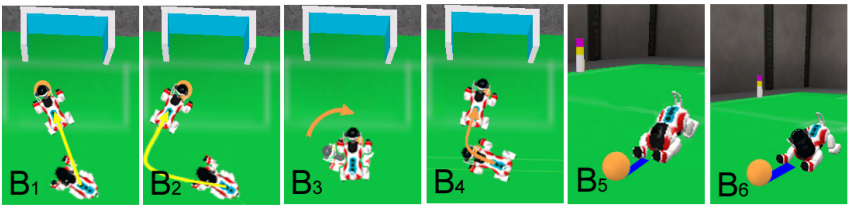

Fig. 1. The six behaviours that can be used in the attacking task

stability of the ball approach are mainly characterized by the **eleven** *walking gait parameters* Θ_{WG}, which define the kinematic characteristics of the walk. The **four** *ball carrying parameters* Θ_{BC} characterize the way the robot slows down and eventually stops near the ball. These parameters also influence the quality of ball control in attacking strategies with no ball carrying (e.g., strategy $\{B_1; B_5\}$). Finally,the robot kicks are generated by a sequence of fixed joint positions, characterized by **nine** *ball kicking parameters* Θ_{BK}.

3.2 Learning to Attack

Here, we present the optimization problem that must be solved in order to improve the attacker performance, based on the above system characteristics. As aforementioned, we present a learning approach that allows for the concurrent search of optimal behaviors and optimal parameters. The first issue deserves some clarification. We represent the attacker strategy P_i with a string of three integers, instead of symbolic expressions. We use the coding function:

$$c : P_i \rightarrow \{\phi_1\ \phi_2\ \phi_3\} \in \mathbf{Z}^3 \qquad\qquad i = 1, \ldots, 12$$

such that: ϕ_1 indicates the *ball approaching* behaviour used in strategy P_i (1 for B_1, or 2 for B_2), ϕ_2 the *ball carrying* behaviour (1 for B_3, 2 for B_4, 0 for no ball carrying), and ϕ_3 the *ball kicking* behaviour (1 for B_5, 2 for B_6). For example, the strategy $\{B_1; B_4; B_5\}$ is coded with the string $\{122\}$.

With this approach, the search for the optimal composition of simple behaviors amounts to a parameter optimization problem in the discrete set $S_\phi = \{1, 2\} \times \{0, 1, 2\} \times \{1, 2\} \subset \mathbf{Z}^3$. In practice, since we consider parameter and behavior learning together, a candidate solution of the optimization problem is:

$$\underline{X} = [\theta_{WG,1}\ \ldots\ \theta_{WG,11}\ \theta_{BC,1}\ \ldots\ \theta_{BC,4}\ \theta_{BK,1}\ \ldots\ \theta_{BK,9}\ \phi_1\ \phi_2\ \phi_3]^T \in \mathbf{R}^{24} \times S_\phi$$

The very large dimensions of the search space, and the system characteristics, suggest the use of the *layered learning* paradigm [9] 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 classic layered learning, we utilize the same learning method for each layer of the hierarchy. Specifically, we can decompose optimization of the attacker task in the following four optimization subtasks (layers):

- L_1: find $\underline{X}_{1,opt} = [\theta_{WG,1}\ \ldots\ \theta_{WG,11}]^T_{opt} \in \mathbf{R}^{11}$ for 'best' walk;
- L_2: find $\underline{X}_{2,opt} = [\theta_{BC,1}\ \ldots\ \theta_{BC,4}]^T_{opt} \in \mathbf{R}^4$ for 'best' ball carrying, given $\underline{X}_{1,opt}$ found by L_1;
- L_3: find $\underline{X}_{3,opt} = [\theta_{BK,1}\ \ldots\ \theta_{BK,9}]^T_{opt} \in \mathbf{R}^9$ for 'best' ball kicking, given $\underline{X}_{1,opt}$ and $\underline{X}_{2,opt}$ found by L_1 and by L_2;
- L_4: find $\underline{X}_{4,opt} = [\phi_1\ \ldots\ \phi_3]^T_{opt} \in \mathbf{Z}^3$ for 'best' attacking strategy, given $\underline{X}_{1,opt}$, $\underline{X}_{2,opt}$, and $\underline{X}_{3,opt}$, found by the three previous layers.

We note k_i the dimension of $\underline{X}_i$ at each level L_i ($k_1 = 11$, $k_2 = 4$, $k_3 = 9$, $k_4 = 3$). The solution can be obtained as:

$$\underline{X}_{opt} = [\underline{X}_{1,opt} \;\; \underline{X}_{2,opt} \;\; \underline{X}_{3,opt} \;\; \underline{X}_{4,opt}]^T$$

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

$$F(\underline{X}) = k_{WG} f_{WG}(\underline{X}) + k_{BC} f_{BC}(\underline{X}) + k_{BK} f_{BK}(\underline{X})$$

where k_{WG}, k_{BC}, k_{BK} are positive weights indicating the significance desired for each of the three aspects in the learning process, and f_{WG}, f_{BC}, f_{BK} indicate respectively the quality of the walking gait, of ball carrying, and of ball kicking. These functions are derived with heuristics on the robot and ball positions at various stages of the attacking task.

3.3 Learning Techniques

Our objective is to maximize $F(\underline{X})$ in a space of dimension k. This is not trivial, since $F(\underline{X})$ is 'black box', analytical computation of its derivatives is impossible, and all the parameters are box-constrained due to the physical characteristics of the system (we note : Δ_j the range size for each θ_j). Hence, conventional derivative-based optimization methods cannot be utilized, and convergence analysis of a method is impossible. The selected approach must handle non differentiable search space, have high convergence rate, and be resistent to noise in $F(\underline{X})$. Many algorithms possess these characteristics. In particular, we will focus on three different machine learning algorithms: Genetic Algorithms, Nelder-Mead Simplex Method, and Policy Gradient.

In *Genetic algorithms* [2], a *population* of q parameter sets (*individuals*) is used to find a solution for the optimization problem. To evolve a new population from the tested one, the q_e best individuals are preserved (*elitism*) and the remaining individuals are generated by applying *crossover* (q_c individuals) and *mutation* (q_m individuals) operators.

The *Nelder-Mead simplex algorithm* [7] explores a search space of dimension k by moving a *simplex* of $v = k + 1$ vertices via four possible geometrical transformations: reflection, expansion, contraction, and shrinkage. From an initial parameter set ${}^0\underline{X}$, the other ${}^l\underline{X}$ vertices ($l = 1, \ldots, k$) of the first simplex are generated as: ${}^l\underline{X} = {}^0\underline{X} + [0, \ldots, \pm\zeta_l, \ldots, 0]^T$.

The *Policy Gradient algorithm* has been used for robot learning in [6]. From an initial parameter set ${}^0\underline{X}$, p randomly generated *policies* ${}^m\underline{X}$ ($m = 1, \ldots, p$), are evaluated, such that: ${}^m\underline{X} = {}^0\underline{X} + [\rho_1, \ldots, \rho_k]^T$, and each ρ_j is chosen randomly in the set $\{+\epsilon_j, 0, -\epsilon_j\}$. Each ${}^m\underline{X}$ is grouped into one of three sets for each j: $S_{+\epsilon,j}$, $S_{0,j}$ or $S_{-\epsilon,j}$ depending on the variation applied to its j^{th} parameter. $-\epsilon_j$. The average objective functions $\bar{F}_{+\epsilon,j}$, $\bar{F}_{0,j}$, and $\bar{F}_{-\epsilon,j}$ are computed for $S_{+\epsilon,j}$,

$S_{0,j}$ and $S_{-\epsilon,j}$. These are used to estimate the *gradient*, which is then scaled by a *step-size* η, and added to ${}^0\underline{X}$, to begin the next iteration.

4 Experimental Results

Here, we report the experimental results obtained by applying the proposed learning methodology in the described robot soccer scenario. More specifically, we comment on three results: 1) the effectiveness of the incremental approach; 2) the comparison among the three learning methods 3) the effectiveness of using a 3D simulator for speeding up the learning process. We have executed the incremental learning approach presented in the previous section, but without considering the fourth layer L_4, since this would require additional input to the system. In fact, selecting the best strategy depends on other variables, such as the position of the robot and of the ball with respect to the target goal, the position of other robots in the field, etc. For layer L_4 it is necessary to learn a function that maps the current situation with a suitable strategy. This aspect needs to be further investigated and it is beyond the scope of this paper. In practice, we applied incremental learning, focused on layers L_1 to L_3, for optimizing the strategy $P_1 = \{B_1; B_5\}$, i.e., the robot approaches the ball as fast as possible, and kicks it forward, without grasping it, with fixed head pan. Learning this task has been initially developed and configured within the 3D AIBO simulator [10] embedded in USARSim[2], before experimentation on the real robot.

Let us briefly outline the configurations of the three learning algorithms. For the genetic algorithm, we use: $q = 10$, $q_e = 1$, $q_c = 6$, $q_m = 3$. *Selection* of individuals from the original population is based on the popular roulette wheel scheme, and mutation is obtained by altering the j^{th} parameter with an offset chosen randomly in the set $[-0.2\Delta_j, +0.2\Delta_j]$. For the Nelder-Mead algorithm, at each layer L_i: $v = k_i + 1$, and $\zeta_l = \pm 0.2\,\Delta_j$, with the sign of ζ_l chosen randomly. For the policy gradient, we use for the three layers: $p_1 = 8$, $p_2 = 4$, $p_3 = 6$, $\epsilon_j = 0.1\ \Delta_j$, $\eta = 3$. The same initial parameter set ${}^0\underline{X}$ is used for starting each learning technique. Since there is significant noise in each experiment, each set of parameters is evaluated three times, and the resulting fitness *evaluated* for that set, is computed by averaging over the three experiments. For each layer, and each learning technique, we terminate learning after $10\,k_i$ *evaluations* (e.g. for L_1, 110 evaluations, i.e., 330 experiments). We choose to use the same amount of learning time, since this is usually a given specification in learning problems.

The results of incremental learning are shown in Fig. 2(a): Nelder-Mead slightly outperforms GA and PG. A similar plot is shown in Fig. 2(b), where we ran the same number of evaluations by learning all 24 parameters at the same time. Comparison between the two plots confirms the quality of the incremental approach: for instance, for the GA, the final fitness obtained by the incremental approach is 34% higher than that obtained by learning all parameters together.

Other experiments were carried out to show how the optimal low-level parameters are strongly related to the desired behavior. To emphasize this aspect, we

[2] `usarsim.sourceforge.net`

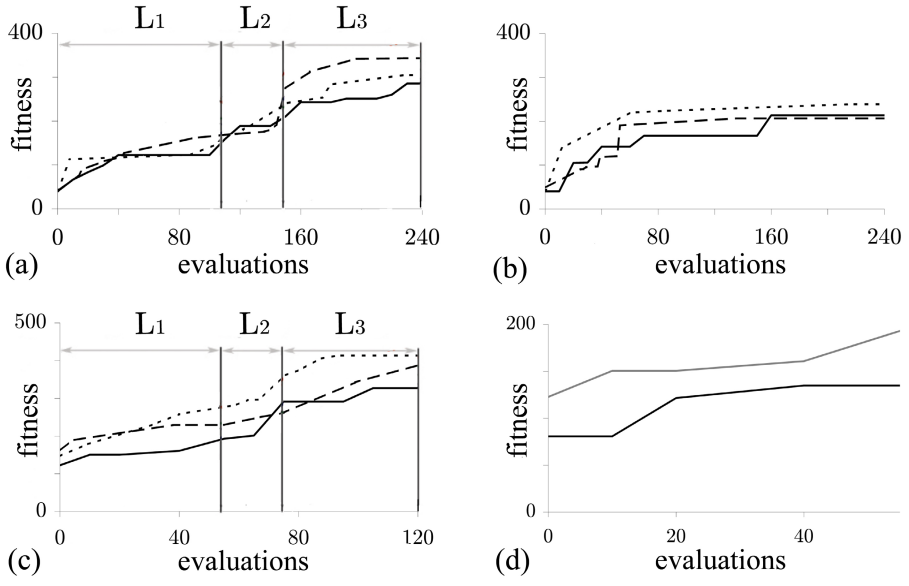

Fig. 2. Fitness values at each evaluation: (a) incremental learning with USARSim, (b) learning all parameters with USARSim, (c) incremental learning with AIBO (GA solid, NM dashed, PG dotted) (d) comparing GA for L_1: with initial population derived from USARSim (grey), and with random initial population (black)

used the same learning configuration used for optimizing strategy P_1, to optimize strategy $P_2 = \{B_1; B_3; B_5\}$ (the robot approaches the ball as fast as possible, rotates while grasping it, and kicks it with fixed head pan). This experiment was carried out with the genetic algorithm, in USARSim: starting from the optimal walking gait parameters $\underline{X}_{1,opt}$ learned for the previous strategy, we proceeded with the other two levels to derive $\underline{X}_{2,opt}$ and $\underline{X}_{3,opt}$. Comparison between the optima learned for P_1 and P_2 showed major differences. Specifically, for P_2, the robot must slow down farther away from the ball (parameters $\theta_{BC,1}$ and $\theta_{BC,2}$ are different in the two cases) and the head movement for kicking the ball (which in P_2 has been grasped) is different and depends on parameters $\theta_{BK,7}$ and $\theta_{BK,8}$. These results outline the dependency of the behavior parameter sets on the chosen strategy.

After having configured the algorithm for learning strategy P_1 in USARSim, we ported it on the real robot. The initial parameter set for each technique is the set derived at the end of simulator optimization. This time, for each layer, and each learning technique, we terminate learning after $5\,k_i$ evaluations. The results of the incremental learning algorithm are shown in Fig. 2(c). On AIBO, policy gradient slightly outperforms the two other approaches.

To emphasize how the use of a 3D simulator speeds up the learning process on real robots, we ran another experiment where the GA was used to learn L_1 for the same strategy P_1, starting from a random population, different from the one derived at the end of USARSim optimization. The results of this experiment

are shown in Fig. 2(d), where the fitness of GA walking gait learning starting from different populations are plotted: the figure clearly shows the advantage of the simulator for deriving the GA initial population.

5 Conclusions

In this paper we presented a layered-like approach for learning optimal parameters, and strategy selection. We compared three different methods in the different layers: genetic algorithm, Nelder-Mead, and policy gradient. Moreover, we showed how the use of a 3D simulator speeds up robot learning. The proposed learning methodology has been applied to the soccer attacking task, on both simulated and real robots, and it has shown a notable improvement in the performance of the robot basic behaviours. The main results of the experiments are: the utility of the layered approach for this complex scenario, and the effectiveness of the 3D simulator for configuring the learning algorithms before porting on the robot. Incremental learning has been executed without considering the strategy selection layer. In fact, this depends on the 'game situation' (e.g., positions of robot and ball with respect to the goal). Learning a function that maps the game situation with a suitable strategy will be the object of further work.

References

1. Arsenio, A.M.: Developmental learning on a humanoid robot. In: IEEE International Joint Conference on Neural Networks, Budapest (2004)
2. Back, T.: Optimization by means of genetic algorithms. In: 36th International Scientific Colloquium, pp. 163–169 (1991)
3. Chalup, S.K., Murch, C.L., Quinlan, M.J.: Machine learning with AIBO robots in the Four-Legged League of Robocup. IEEE Trans. on Systems, Man and Cybernetics, Part C (2006)
4. Millan, J.d.R.: Rapid, safe, and incremental learning of navigation strategies. IEEE Trans. on Systems, Man and Cybernetics, Part B 26(3), 408–420 (1996)
5. Fidelman, P., Stone, P.: The chin pinch: A case study in skill learning on a legged robot. In: Proc. of 10th International Robocup Symposium (2006)
6. Kohl, N., Stone, P.: Machine learning for fast quadrupedal locomotion. In: The 19th National Conference on Artificial Intelligence (2004)
7. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder Mead simplex algorithm in low dimensions. SIAM Journal on Optimization 9, 112–147 (1998)
8. Martinson, E., Arkin, R.C.: Learning to role-switch in multi-robot systems. In: Proceedings of International Conference on Robotics and Automation (2003)
9. Stone, P., Veloso, M.: Layered learning. In: Lopez de Mantaras, R., Plaza, E. (eds.) ECML 2000. LNCS (LNAI), vol. 1810, pp. 369–381. Springer, Heidelberg (2000)
10. Zaratti, M., Fratarcangeli, M., Iocchi, L.: A 3D simulator of multiple legged robots based on USARSim. In: Proc. of 10th International Robocup Symposium (2006)