Learning Model-Free Motor Control

Alejandro Agostini and **Enric Celaya**[1]

Abstract. Some robotic tasks require an accurate control to follow the desired trajectory in the presence of unforeseen external disturbances and system parameters variations. In this case conventional control techniques such as PID must be constantly readjusted and a compromise solution must be adopted. This problem can be avoided using a learning process that automatically learns the appropriate control law and adapts to ongoing system variations. But a drawback of many learning systems is that they are not effective for non-toy problems. In this paper we present the results obtained with a categorization and learning algorithm able to perform efficient generalization of the observed situations, and learn accurate control policies in a short time without any previous knowledge of the plant.

1 INTRODUCTION

In legged robots, unforeseen external disturbances constantly appear when a leg contacts the floor or collides with an object and usually system parameter variations take place. Therefore, complying with the mentioned control requirements with a standard control method such as PID, that requires a precise tuning for every particular situation, is a very difficult task. More sophisticated, model-based forms of control can be used [1], [2], [3], but for the kind of problems we are considering, explicit modeling is often impossible since the actual working conditions are unknown.

A promising alternative is to use an automatic learning system that uses previous experiences to improve its performance with time without any previous knowledge of the model of the plant. Learning from experience is the subject of Reinforcement Learning (RL) [4]. It is well known that the application of RL to non-toy problems, suffers the problem of the curse of dimensionality [5]. To allow the application of learning in complex tasks as the one we are considering here, a new approach has been devised [6] that takes advantage of an environment property called categorizability. Categorizability means that from all the features of the environment that must be taken into account to predict the results of executing an action in any situation only a reduced number of them become relevant to predict the result of an action in any particular situation.

In this work we apply the categorization and learning algorithm to the problem of the trajectory tracking control of a DC motor.

2 THE CL ALGORITHM

In this section we summarize the fundamental aspects of the CL algorithm. For a more detailed explanation see [6] and [7].

It is assumed that the world is perceived through a set of n binary feature detectors f_i i=1...n. A partial view of order m, denoted by $v(f_{i1},..., f_{im})$, is defined as a virtual feature detector that becomes active when its m component feature detectors are simultaneously active. For each partial view v and for each action a three values are associated:

- $q_v(a)$, estimates the average discounted reward.
- $e_v(a)$, weighted average absolute error of the prediction. It is used to define the confidence interval $I_v(a)=[q_v(a)-2e_v(a), q_v(a)+2e_v(a)]$.
- $i_v(a)$, confidence index, estimates how much action a has been tried when v was active resulting in a q value inside $I_v(a)$.

As in the usual Q-learning algorithm, we must determine, for each situation, the action that maximizes the expected q value. The problem in our case is that in a given situation we may have many different predictions for the same action: one for each active partial view. To address this problem we define the winner partial view as,

$$winner(V,a) = \arg\max_{\forall v \in V}\{\rho_v(a)\cdot c_v(a)\} \tag{1}$$

where V is the set of active partial views, $\rho_v(a)$ is a decreasing function of the $e_v(a)$, and $c_v(a)$ is an increasing function of $i_v(a)$. In this way, the q prediction for an action in a given situation will be obtained from the winner partial view for this action.

To get an actual q prediction from the winner partial view, two sources of indeterminacy must be considered. A first prediction is obtained as a random sample with uniform probability in $I_v(a)$. Then a noise term is added that depends on the confidence of the estimation $c_v(a)$. Note that this strategy implements an adaptive form of exploration.

After the execution of an action, a reward r is obtained and a new situation V is perceived. The actual q obtained from the execution of action a can be computed as:

$$q = r + \gamma . \max_{\forall a}\{q_v(a) \mid v = winner(V,a)\} \tag{2}$$

where γ is the discount factor. This information is used to update the estimated values for the executed action of all partial views that were active in the last situation. The $q_v(a)$ is updated using the same rule as Q-Learning for stochastic systems. The $e_v(a)$ value is updated with identical schema. In both cases the factor (1-$c_v(a)$) is used as the learning coefficient to enable faster adaptation in the values of those partial views with lower confidence. Finally, each confidence index $i_v(a)$ is increased by one if the actual q value lies in the predicted interval $I_v(a)$ and decreased otherwise.

If the absolute difference in the prediction is higher than a user defined amount δ the prediction of the q value is considered wrong and τ new partial views are created. New partial views are created by combination of two already existing partial views, randomly chosen among those that were active in the last situation.

[1] Institut de Robòtica i Informàtica Industrial, Barcelona, Spain email: {agostini, celaya}@iri.upc.es

The number of partial views in the system is limited to a threshold μ. To comply with this threshold, it is necessary to remove partial views when its number grows above μ. There are two different elimination criteria: redundancy and utility. A partial view is considered redundant when its reward predictions are too similar to the reward predictions of any of the partial views composed by a subset of its features [7]. On the other hand, the utility of a partial view is used to eliminate partial views that appear to be less useful for the system. Utility is defined as:

$$u(v) = \max_{\forall a} \left\{ \frac{\rho_v(a)}{c_v(a)} \right\} \tag{3}$$

3 THE CONTROL PROBLEM

The control problem consists in learning to follow a randomly generated trajectory for a rotational actuator of a legged robot. Two motors were modeled for the control task, the Maxon 118800 and 118769 with different power ratings [8]. The actions used for the control task are motor input voltages.

Since this problem has continuous features, a discretization is required. The features considered are the angular speed and acceleration, the difference between the actual and the desired speed, and two features for the difference between the current and the desired position, one with higher resolution than the other in order to reduce the exploration needed. To increase the resolution near the trajectory, a logarithmic scale is used for those features involving differences with respect to the desired trajectory. We consider 40 discretization intervals for each feature. Actions are discretized using 16 equal segments.

We define a reward function that takes into account, not only the distance to the trajectory but also the angular speed of the motor as well as the angular speed of the reference trajectory,

$$\theta_{target}(t) = \frac{\left[\theta_M(t-\Delta t) - \theta_T(t)\right] + \dot{\theta}_T(t)\Delta t}{2} + \theta_T(t) \tag{4}$$

$$r(t) = -\left|\theta_M(t) - \theta_{target}(t)\right| \tag{5}$$

where θ_{target} is the current desired angular position, θ_M is the motor angular position, and θ_T is the trajectory angular position.

4 RESULTS

Fig. 1 shows the square error of each system. The controls reached by the CL algorithm for motor 118800 and 118769 are comparable with those of the PID tuned control systems. This is a significant result considering that the learned set was trained using 25000 iterations that correspond to only 20 minutes of motor operation. In these cases, the number of partial views was limited to only 500. This is very low compared with the large number of possible states that a conventional reinforcement learning algorithm would need. This fact demonstrates the high categorization reached by the CL algorithm. It is important to mention that in order to reach a good control performance with the PID for the motor 118769 it is necessary to re-tune the control system.

5 CONCLUSIONS

Learning techniques could be a good solution for control in highly variable environments and under system's parameters variations, but the use of techniques such as RL are problematic due to the extremely large computational resources and convergence times they need in realistic control problems.

The CL algorithm is able to reduce both the storage needs and the convergence time to make possible the learning of control policies whose performance can be compared with optimally tuned PID control. Moreover, the CL algorithm does not need any previous knowledge of the plant and can learn a good control policy even in the presence of large plant parameters variations.
All these arguments make the CL algorithm learning system very promising to the control task for robot locomotion.

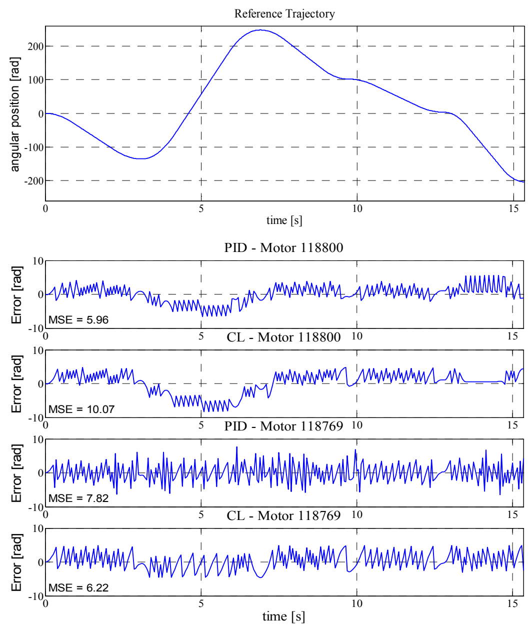

Figure 1. Reference trajectory and square control errors of each control system

ACKNOWLEDGEMENTS

This work has been partially supported by the Ministerio de Ciencia y Tecnología and FEDER, under the project DPI2003-05193-C02-01 of the Plan Nacional de I+D+I.

REFERENCES

[1] L. De S. Martins-Filho, J. Silvino, P. Presende, T. Assunçao, 'Control of robotic leg joints – comparing PD and sliding mode approaches'. *Proc. Of the Sixth International Conference on Climbing and Walking Robots*, Catania, Italy, September 2003, pp.147-153.

[2] L. Crawford, S. Shankar Sastry, 'Learning Controllers for Complex Behavioral Systems', *UC Berkeley ERL Memo M96/73*, 1996.

[3] E. Burdet, J. Luthiger, 'Three learning architectures to improve robot control: A comparison', *Proc. MLnet Familiarization and Third European Workshop on Learning Robots*, Heraklion, Greece, April 1995, pp. 28-29.

[4] R. Sutton and A. Barto. *Reinforcement Learning. An Introduction*. MIT Press, 1998.

[5] R.E. Bellman. Dynamic Programming, Princeton University Press. Princeton.1957.

[6] J.M. Porta and E. Celaya. 'Learning in Categorizable Environments'. *Proc. of the Sixth International Conference on the Simulation of Adaptive Behavior*, Paris, September 2000, pp.343-352.

[7] A. Agostini and E. Celaya. 'Learning in Complex Environment with Feature-Based Categorization'. *Proc. 8th Conference of Intelligent Autonomous Systems*, Amsterdam, March 2004, pp. 446-455.

[8] Maxon Motor, *High Precision Drives and Systems*, Maxon Interelectric AG, Switzerland.