REINFORCEMENT LEARNING AND AUTOMATIC CATEGORIZATION

Josep M Porta and Enric Celaya

Institut de Robòtica i Informàtica Industrial (UPC-CSIC), c/Gran Capità 2-4, Edifici NEXUS, Planta 2
08034, Barcelona (SPAIN) T.+34 93 401 57 91 F.+3493 401 57 50 E-mail: {jporta, celaya}@iri.upc.es

Abstract - *The categorization process defines sensor and action categories from elementary sensor readings and basic actions so that the necessary elements for solving a task are correctly perceived and manipulated. In reinforcement learning, a previous categorization process is needed to define sensor and action categories with special requirements that we analyze in this paper and that, in general, are difficult to achieve, specially in complex tasks such as those that arise when working with autonomous robots. We show how these special requirements should be relaxed and we sketch a reinforcement learning algorithm that uses a less restrictive form of sensory categorization than existing algorithms. Additionally, we show how a given sensory categorization can be improved so that it better fits the demands of the previous algorithm.*

1. INTRODUCTION

Continuos improvements in areas such as Mechanical Engineering and Micro-Electronics have given us the possibility of building autonomous robots with increasingly sophisticated sensorial and motor systems. The control of this kind of robots to accomplish complex tasks in dynamic environments is one of the challenges of Artificial Intelligence but we are far from achieving it.

In the last years, some authors [2] have argued that traditional Artificial Intelligence approaches based on high level reasoning and planning are not adequate for the type of problems that arise when controlling a real robot in real time, and new control architectures have been proposed. The use of these new architectures allowed to achieve complex tasks with controllers based on simple principles [1], [5]. But, even in these new paradigms, the programming of a robot is not free of problems. Behind each success there is usually a programmer that has spent lots of hours designing the controller, developing each one of its modules and adjusting their parameters and interactions until the desired behavior is obtained. If we want to address more complex tasks and environments, this programming process must be alleviated, and the application of machine learning techniques is a promising way to explore. The idea of automatic learning of robot controllers has been present in the new robot control architectures from the very beginning [6], [10], [11], [12].

A learning paradigm can be classified according to the amount of information that the designer directly provides to the learning agent[1]. At one end of this continuos spectrum (closer to the manual programming) we find the supervised learning approaches in which the designer informs the agent about which are the adequate actions (with respect to the current task) to be executed in some specific situations and the objective of the learning algorithm consists in generalizing the given information to find out the correct actions in all the possible situations. At the other end of the learning paradigms, there are the *reinforcement learning* systems [8]. In these systems, the designer only informs the agent when the task has been successfully completed. Between the two extremes of the classification, there is a variety of learning systems that the programmer can use depending on her knowledge of the task and on how easy is it to transfer this knowledge from the designer's point of view to the robot's point of view. In many problems concerning autonomous robots the reinforcement learning paradigm is the only adequate one.

Reinforcement learning has been extensively studied since the origins of Artificial Intelligence and even before. However, as it happened with the traditional Artificial Intelligence architectures, learning paradigms must be adapted to be successfully applied to autonomous robots. Since now, existent reinforcement learning algorithms have been mainly focused on how to find the correct links between perception and action without paying much attention on how perception and action must be pre-processed so that relevant mappings between them can be established. In general, current reinforcement learning algorithms depart from the basis

[1] There exits also unsupervised learning approaches in which no information is given to the learning agent but they are not task-purposive.

0-7803-5670-5/99/$10.00 ©1999 IEEE

that states for solving the task can be identified and that adequate actions to go from one state to another are available to the agent (see [14] page 61 for a good explanation on this assumption). The definition of these states and actions is a type of *categorization*. This process is in charge of the programmer and is often implicit in the application of the reinforcement learning algorithms. The categorization process can be the most difficult stage of the solution of a task (the proper interpretation of the sensor readings can transform an apparently complex task in a simple one) so, if it is done by the programmer, we could be systematically confronting our reinforcement learning algorithms with the easiest part of the problem. This is specially true when facing difficult tasks or environments with complex agents (such as autonomous robots). In this case, determining the state from the sensor readings and the appropriate combinations of elementary actions becomes so difficult that the application of reinforcement learning algorithms is limited to simple problems.

In the next section we analyze in more detail the special properties of the sensor and action categories required for the correct application of classical reinforcement learning algorithms. We also show how such properties become a problem when these algorithms want to be applied to complex agents such as autonomous robots. In section 3, we sketch an extended reinforcement learning algorithm able to deal with a less restrictive form of sensory categorization than that required by the available algorithms, and in section 4 we describe how this categorization can be automatically improved using the information given by the reinforcement signal. Finally, in section 5 we extract some conclusions of our work and outline future ways in which our system should be extended to cope with more general learning problems.

2. THE REINFORCEMENT LEARNING ASSUMPTIONS

The reinforcement learning paradigm has been extensively studied in areas such as Ethology, Automatic Control, and more recently Artificial Intelligence. A general description of a reinforcement learning situation includes the following elements:

- A definition of the task: The only strictly needed information about the task is a signal, called the *reinforcement signal*, that becomes active when the task is achieved. If more information about the task is available (as for instance, correct or incorrect actions in some situations, or necessary preconditions for achieving the task), it can be added to the reinforcement signal to help the learner.
- An environment in which to accomplish the task. Depending on the kind of environment (static, dynamic,...) the resolution of the task can be less or more difficult.
- An agent that must complete the task. This agent can perceive the environment and execute actions. The solution of a task consists in determining actions for each situation so that the agent collects as much reward as possible.

The first attempts to formalize this framework were at a high level of abstraction, and the resulting formalization was not as general as the previous description. The main hypothesis on which this formalization relies is to consider the environment as a state machine controlled by the agent; so that the interaction between the environment and the agent is arranged in a two step loop: In the first step, the agent perceives the state of the environment and, in the second one, it performs an action that produces an immediate change in this state according to a well defined transition probability. This hypothesis was quickly accepted by the Artificial Intelligence researchers coming from fields such as automated reasoning or planning, because they also used to tackle problems at a high abstraction level, in which states and actions to move from one state to another were easy to define (if not directly present in the definition of the problem). So, the reinforcement learning algorithms developed in Artificial Intelligence heavily rely on the above hypothesis and as far as it is not fulfilled, they do not work properly. However, considered from the robotics point of view, the above hypothesis is not realistic at all since in a dynamic environment it is not reasonable to suppose that the global state of the environment is controlled by the robot actions. At most we can assume that part of the state is controlled by the robot but obviously there are thousands of events that occur absolutely out of the robot control. This disagreement between the usual reinforcement learning hypothesis and robotics reality makes existing reinforcement learning algorithms very difficult to be used in robotic environments.

In the following subsections, different aspects of the reinforcement learning formalization hypothesis are analyzed in detail from the point of view of the application of reinforcement learning in a robotic task. This revision will help us to settle the basis on which to describe new algorithms more useful for being applied to robots than the existing ones.

2.1 Global State Identification

The reinforcement learning hypothesis of considering the environment as a state machine controlled by the agent, is usually accompanied by the assumption that the agent has direct knowledge of the state of this machine. In a robotics application this supposes that the robot must be able to identify the global state of the environment. But this objective can hardly be achieved by an agent able to perceive only part of its

surroundings. Despite this limitation in robot perception, the current sensor readings are often taken as the present state of the environment. The robot is supposed to change this state using the elementary actions it is able to execute. Unfortunately, this simple solution does not work properly in most autonomous robot applications. The reason is that sensor readings and elementary actions conform a too large search space [15], and that, in a complex autonomous robot, it is almost impossible to consistently predict the next sensor readings after the execution of an elementary action. This is in contrast with the assumptions underlying the reinforcement learning algorithms, so that, with this definition of states and actions, they can hardly accomplish even the simplest tasks when applied to autonomous robots.

A usual attempt to solve this problem consists in defining *feature detectors*. A feature detector can be devised as process that identifies special combinations of sensor readings. Feature detectors are usually defined by the programmer attending to special characteristics of the environment that the robot is able to perceive. Departing from a set of feature detectors, the state of the system is usually defined as the current combination of active and non-active feature detectors. The high level of abstraction of the feature detectors makes the transitions between states based on them more predictable than when states are defined from elementary sensor readings. For this reason it is more reasonable to apply the traditional reinforcement learning algorithms using states defined in this way.

The definition of feature detectors from sensor readings is usual in robotics. For instance, in [11] sensor readings are conveniently grouped prior to the definition of states both manually and through an automatic process (thus, infrared sensor readings are clustered in two classes using programmer defined criteria, so that only two situations for detected objects can be distinguished: near and far).

However, what is important to be stressed is that the concept of *"global state of the system"* is something that has little sense to be defined in the real world. The only thing we need in this environment is to get enough information to determine the agent actions and this do not necessarily imply to observe the "global state of the system". But even this partial perception of the world is very difficult to be achieved by an agent with a limited sensorial apparatus and the definition of feature detectors is nothing but an approximation to the necessary information to act correctly.

The definition of feature detectors is the result of a categorization process and, as sated before, it could be the most difficult part of the solution of a task so an automatic process to define them (as the one we present in section 4) would be helpful.

2.2 Independent States

The states of the system (that, as explained in the previous section, are supposed to be accessible to the agent) are usually thought as being absolutely independent between them. So, the information gathered about the effects of an action in one state can not be transferred to other states. This is a really general assumption that makes reinforcement learning algorithms completely independent of the environments on which they are applied. But the drawback of this assumption is that the algorithms do not work efficiently in large spaces of states since they have to discover the information concerning each one of the states independently. In a robotics application this assumption is unnecessary since in general states are related between them because they correspond to a physical reality with a great degree of continuity. For example, the actions to be executed when a robot is close to a wall could be quite similar to those to be executed when it is *very* close to the wall. So in a robotics learning application it is sensible to use the information about state relations to accelerate the learning process.

One way to transfer information between states is to make the reinforcement learning algorithms to work at the feature detector level instead of at the state level. The reason is that if we discover information about the effects of an action when a feature detector is active, this information will be useful for all the states that include this feature detector. Observe that if we use feature detectors to define a state and we apply a standard reinforcement algorithm on this state definition (which is the usual solution taken in reinforcement learning) we are not associating information to each feature detector and so we are not transferring information between states.

The feature detector framework is nothing else than a form of sensory categorization alternative to that consisting in defining state identifiers. We have shown that in a robotic task, it is more reasonable to work with a more basic categorization than that demanded by the classical reinforcement learning algorithms. So, even though the concept of state is useful when defining an abstract model of the reinforcement learning, when it is applied to robots it is more useful to work at the level of the basic state constituents.

In section 3 we define a new reinforcement learning algorithm that works at the feature detector level and that can be more naturally applied to robots than available algorithms.

2.3 Independent Actions

The model of reinforcement learning presented above devises actions as a list of options from which the agent picks the most appropriate for each state. At each

moment the agent only performs one action from the list. In an autonomous robot with many degrees of freedom this usually implies the definition of a large set of complex actions that in many cases is drastically reduced by the programmer for efficiency reasons[2]. However, in general, these complex actions are defined combining independent simpler actions that involve only a subset of the motor apparatus of the robot.

As in the case of states, when confronting a robotic task it seems more sensible to work at the level of the constituents of the global actions. Working at this more basic level allows us to get information about the effect of each basic action. The drawback of working at this more basic level is that we have to deal with the execution of many actions in parallel. This implies the definition of a mechanism for discerning which actions can (or even should) be executed at the same time.

If we decide to work at the elementary action level, then we are changing the action categorization from what the traditional reinforcement learning algorithm demands to one that is more natural when working with robots.

2.4 Environment Controlled by the Agent

In the reinforcement learning formalization, it is assumed that the environment in only modified by the robot actions. This means that the agent only needs to observe the state of the environment when the action in execution is completed.

Obviously, this assumption does not hold any more if the state of the environment is not completely controlled by the acting agent. In this case the state can change at any moment for reasons not related with the agent actions and this could perfectly occur in the middle of an action execution. For this reason, when applying reinforcement learning to robots, it seems reasonable to observe the state of the system continuously since it can change at any time.

This can be seen as another change in the typical action categorization of reinforcement learning. Robot environments demand the learning algorithms to be able to deal with actions that last more than one step considering the time unit to be the period at which the state is observed (that can depend on the sensor updating frequency).

2.5 The Immediate Effect of Actions

Another assumption present in the reinforcement learning algorithms concerns the moment at which the agent observes the effect of its own actions. Traditionally it is assumed that the effect of an action is perceived immediately after its execution. But when working with a robot in the real world, this assumption is not valid. In this case, the robot actions can start processes whose effects are not evident for the robot itself just after the action execution. The action effects can be arbitrarily delayed from the moment at which the action was performed. Observe that the delay in the effect of the actions is possible even if the state of the system is only modified by the agent actions (what was the assumption analyzed in the previous section). If we are trying to solve a problem using actions with possibly delayed effects (which is a usual case in robotics), then we should use an algorithm that can cope with this feature. The available reinforcement learning algorithms are not adequate for these cases.

[2] For example, in [11], only five actions (forward, small turn left/right, and large turn left/right) are considered despite the fact that the robot could perform many other movements.

3. AN EXTENDED REINFORCEMENT LEARNING ALGORITHM

In this section we present a reinforcement-based algorithm in which some of the requirements imposed by the traditional reinforcement learning algorithms on the initial categorization of the problem are eliminated.

From all the assumptions present in the classical reinforcement learning framework commented in the previous section, we start relaxing the sensory categorization: our algorithm works with a sensory categorization based on feature detectors instead of using a state identification function. As explained before this will make our algorithm more suitable to be applied to robot tasks in which feature detectors are more easily defined than state identification functions. Additionally if we extract information about the effects of actions when a specific feature is active, then this information will be used in all those states that include this feature. In this way it is reasonable to expect our algorithm to work more efficiently than existing algorithms.

How the feature detectors are implemented, and the characteristics of the special event they signal is something that does not matter for the algorithm we describe now (it will be important for the algorithm described in the next section). Now the only thing that is relevant is that we can test whether or not a feature detector is active.

We keep all the restrictions imposed by the existing algorithms on action categorization, so the programmer should provide our algorithm with a list of actions that are supposed to be sufficient to accomplish the task. These actions will be considered to be the only font of variation in the environment and to have immediate effects.

The objective of the algorithm is to determine a policy that consists in finding the best action to be executed given a set of active feature detectors.

As described before, lots of feature detectors can be active simultaneously denoting interpretations of different subsets of the perception of the robot, or even of the same subset of the perception but with alternative criteria or goals. Each active feature detector assigns a different utility to the execution of each action. The action to be actually executed must be chosen from all these individual proposals attending to criteria related with the accumulated utility per action and the confidence in this utility measure. The process in charge of this selection task is called the *action selection mechanism*[3] and is supposed to be provided by the programmer. Different action selection mechanism will produce different behaviors of the algorithm.

The estimated utility of executing and action (j) when a certain feature detector (i) is active is stored in a table (Q(i,j)). A subset of this table (that corresponding to the active feature detectors) is the information provided by our algorithm to the action selection mechanism. Each entry to this table is updated when appropriate using a rule similar to that used in other reinforcement learning algorithms. When all the values stored in the table are stable, our algorithm has converged to a solution for the task at hand.

The main loop of our algorithm consists in:

1.- For each feature detector i and action j
Initialize Q(i,j)=0
2.- Do forever
- *X = current set of active feature detectors*
- *a = Action Selection(X,Q)*
- *Execute a*
- *Update the utility of action a for all the feature detectors in X*

The update of the utility works as follows:

R = reward received after the execution of a
Y = set of active feature detectors after a
b = Action Selection(Y,Q)

$$V = \sum_{i \in Y} Q(i,b)$$

$\forall i \in X$

$$Q(i,a) = Q(i,a) + \alpha(p_i[R + \gamma V]) - Q(i,a))$$

$$\text{with } p_i = \frac{Q(i,a)}{\sum_{i \in X} Q(i,a)}$$

where α (learning rate) and γ (utility discount) are the commonly used parameters in the reinforcement learning algorithms [8].

[3] This mechanism is implicit in other reinforcement learning algorithms but it has been explicitly studied in the field of autonomous robot architectures.

Observe that if only one feature detector is active at a time and the action selection mechanism always picks the action with maximum expected utility then, the above algorithm is completely equivalent to the well known Q-Learning [16].

4. A CATEGORIZATION IMPROVEMENT ALGORITHM

In the previous section we have described a reinforcement based algorithm that can cope with a form of sensory categorization less restrictive than that required for other reinforcement learning algorithms. Obviously, as occurs with many algorithms, the efficiently of the presented algorithm clearly depends on the quality of its inputs (the feature detectors set). An implicit assumption of the presented algorithm is that the feature detectors are independent between them so that the best action to be performed can be determined (at least some times) attending to only one (or a few) features. If at every moment all the active features have to be taken into account to determine the optimal action, then our algorithm looses its advantage over existent algorithms because this means that each state is completely independent of the other states and that no information can be transmitted between them. Fortunately, as noted before, in robotics applications it is usually possible to define feature detectors as those needed by our algorithm. What can happen is that even if it is possible to define the proper feature detectors, the user can not do it because it is too difficult for her to correctly describe the adequate concepts for achieving the task in terms of the robot sensorial apparatus (the so called *frame of reference problem* [13]). In this cases the programmer can only provide the algorithm with a set of approximately correct feature detectors. For this reason, we introduce a mechanism so that the initial categorization provided by the programmer can be improved to facilitate the workings of the reinforcement learning algorithm.

The automatic categorization improvement mechanism embeds the algorithm presented in the previous section in the following way:

1.- Set up an initial set of feature detectors
2.- Run the algorithm of section 3 until it converges to a policy.
3.- Improve the set of feature detectors.
4.- If this set has changed, go to 2.

The search for the adequate feature detectors is done in step number 3 using a genetic algorithm. To describe its workings we first describe its search space (or the space of syntactically correct chromosomes) and in second place we concentrate in the criteria for selecting optimal feature detectors (the fitness function).

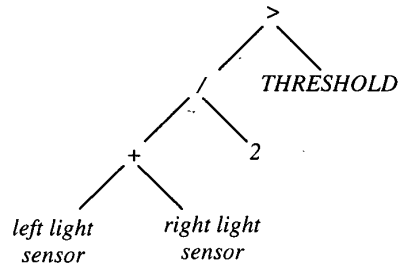
4.1 The Search Space

Feature detectors can be devised as processes that identify special combinations of sensor readings. From the different possible ways to identify combinations of sensor readings (clustering is one of the most typical) we choose a general method that consists in defining binary functions (or predicates) over sensor readings. For instance, the category "*light in front of the robot*" can be defined based on the readings of the front light sensors of the robot as:

(left light sensor+right light sensor)/2>THRESHOLD

This predicate represents a feature detector that will become active if the robot approaches a light.

The search space of the genetic algorithm includes all the syntactically correct predicates definable over constant values, sensors or previously defined predicates. Those predicates can be represented as function trees with sensors or constants in thier leaves and basic operations (from a set provided by the programmer) in their nodes. In the above example, the feature detector is defined using two sensors (left and right light sensors), two constants (2 and *THESHOLD*) and the '+', '/' and '>' operations. The function tree that represents this predicate is:



In other cases other sensors and operations can be used to define the function that represents the feature detector. If some knowledge is available about the environment or the task, the user can reduce the generality of the system by selecting those sensors that can be combined, or restricting the set of applicable operations. This will alleviate the work of the automatic categorization.

This feature detector definition mechanism is very general since almost all the categorizations performed up to this moment in reinforcement learning, either by the programmer or automatically, fit in this framework (the differences between the existent approaches are the functions used to combine the sensors).

New functions that represent new feature detectors can be created using various mechanisms typical of the genetic algorithms:

- Crossover: Part of the function tree of a feature detector is changed with a subtree from another feature detector.
- Mutation: A new function is obtained changing at random one of the parameters of an existing function.

The search space just outlined is unbounded since it is possible to define an infinite number of syntactically different binary functions. Fortunately, it is reasonable to bias the search toward testing first simple functions and use those that seem useful in various cases to build more complex functions. At the beginning only sensors are used to create simple feature detectors but, as good feature detectors are found, more elaborated ones can be built using the previous ones. It is like trying to find the 'atoms' of the correct environment interpretation to build more and more complex 'molecules' using them and other simpler 'molecules'.

4.2 The Fitness Function

The fitness function used by the genetic algorithm to select the best feature detectors for the reinforcement learning algorithm described in section 3 is based on three criteria: environment regularities, robot-environment interaction, and relation with the reinforcement signal.

Environment Regularities

Feature detectors must identify objects or situations interesting for the current task. In the absence of reinforcement related information, however, it is not possible to know whether or not a feature is useful for the current task. What seems sensible to do is to identify objects and situations characteristic of the environment so that, when the reward becomes active, it can be accounted for in terms of these already identified objects which provide information at a higher level than the basic sensor readings. This process of discovering environment related features can be based on the analysis of the activation frequency of different detectors: only those features that become active not too frequently are interesting since when active, they indicate special events. This criterion is based on the assumption that the reward is a special event so only the feature detectors that signal rare events will be useful to identify reward situations.

Robot-Environment Interaction

Those environment regularities detected with the previous criterion that are somehow controllable by the agent actions are more suitable to be used by the reinforcement learning algorithm to choose actions in order to actively get reward. So, it is interesting to stress those couples of feature detectors (f_i, f_j) related by any action. If after the execution of action 'a' when f_i is

active, feature f_j becomes consistently active, both f_i and f_j should be favored.

Relation with the reinforcement signal

Observe that using the two previously described criteria, we are finding feature detectors independently of the information given by the reinforcement signal. The advantage of finding task independent features is that they can be useful for many similar tasks and environments. Additionally, it is interesting to be able to refine the categorization while no reward is obtained since, in this way, when the reward is obtained the reinforcement learning algorithm will use it more efficiently because the utility will be assigned to feature detectors with less probability of being eliminated in future iterations of the categorization improvement process.

The problem of the task independent features is that there are too many of them since there exist lots of possibly interesting objects in the environment to be identified and thousands of ways to do it. So the information provided by the reinforcement signal must be used to filter them. We prefer feature detectors that identify situations in which the execution of a given action produces a high utility either because of the immediate reward received or because the utility of the new situation produced by the action. This utility measure is precisely the one stored in the Q table described in section 3.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have analyzed the assumptions hiden behind the classical formalization of reinforcement learning. This formalization supposes that the environment can be seen as a state machine controlled by the learning agent. This hypothesis implies a set of derived assumptions:

- The learning agent can identify the global state of the environment.
- States are independent and no information can be transmitted between them
- Actions are also independent and no conclusion can be drawn from the execution of an action on the possible effects of other actions.
- Only the agent actions produce effects in the environment.
- The effect of an action is observed immediately after its execution.

These assumptions, when analyzed from the point of view of a robot, become a hard limitation to the application of reinforcement learning since fulfilling all the assumptions is not always possible without changing in important ways the problem to be solved.

To overcome these limitations, we have proposed a new reinforcement-based learning algorithm. Our algorithm can cope with a form of sensory categorization less restrictive than that used by existing algorithms. The kind of categoritzation we use is based on feature detectors (that can be thought as functions that identify special events or objects). Having the information stored separately for each feature detector allows the transmission of information between related states: the information of a feature will be useful in all those states in which the feature is active. Additionally, we have introduced a mechanism based on genetic algorithms to automatically improve the set of feature detectors that the programmer provides to the algorithm. With the combined action of the two algorithms the set of attainable problems using reinforcement learning has been enlarged.

Our system is related with many other disciplines:

- Behavior based controllers: Our algorithms are intended to be used to facilitate the programming of robots. The proposed algorithm based on feature detectors, generates a policy that can be easily translated into a set behavior as those described in [2]. Note that the resulting controller will be reactive only if the operations over the sensor inputs to define feature detectors are reactive. In our intent of extending the limits of the reinforcement learning paradigm we have shown the necessity of introducing an action selection mechanism in this paradigm. As noted before, this is a mechanism already existing in autonomous robot architectures, especially in the reactive ones. This makes us suspect that the parallelism between extended reinforcement learning algorithms and behavior based controllers will be an interesting subject to study. Indeed many authors have tried to analize the relation between the two paradigms. What we are starting to show is that, with the appropriate extension, the two paradigms could be merged.
- Genetic programming: Some authors have used genetic algorithms to evolve programs that perform a task [9]. We also use genetic search techniques by they are only part of our system.
- Multitask learning: The algorithm can generate feature detectors derived from environment regularities and not directly related with the task to be accomplished. These detectors are potentially useful for many tasks in similar environments. This reuse of the experience is also present in an approach to machine learning called multitask learning [4].
- Classifier Systems: Classifier systems [6], [7] aim to develop a correct controller for a given task combining ideas from genetic algorithms and reinforcement learning but they are based on evolving a production system while our algorithm is not.

Taking into account the history of reinforcement learning we can say that our system goes a step further than the usual reinforcement learning algorithms. The first algorithms to learn through reinforcement were developed in the process optimization area and assumed the knowledge of the complete model (probability transitions between states and actions) of the system to optimize. The next generation of reinforcement learning algorithms relaxed this hypothesis in the sense that they do not require the knowledge of the model of the process but as we have noted in this paper, they heavily rely in the correct definition of states and actions. The algorithm we have proposed relaxes some of the assumptions still present in the reinforcement learning framework. Our algorithm can work in cases in which the definition of states is not available at the beginning of the learning process and only feature detectors are available, so we will be able to solve problems not solvable up to now. This set of new attainable problems includes interesting cases such as those posed when trying to learn a complex task with an autonomous robot. Additionally, our algorithm is able to generalize situations and this increases its efficiency compared with existent algorithms.

The algorithms presented in this paper relax only one of the assumptions present in the reinforcement learning formalization. Obviously other algorithms (or extension to the presented one) could be devised so that more assumptions are eliminated. These new algorithms will be even more applicable to robots enlarging the potential uses of the reinforcement learning framework.

Our final objective is that of learning an autonomous robot controller to accomplish a complex task in a real environment without any previous pre-process of neither sensors nor actions. This, as [3] said, is the most complex thing one can attack in the area of learning within the field of autonomous robots and probably we are still far from solving it. However, it is an objective that must be faced if we want to accomplish increasingly difficult tasks using increasingly sophisticated robots. To confront this challenge, new reinforcement learning algorithms and automatic categorization mechanisms like the ones proposed in this paper will be indispensable.

ACKNOWLEDGEMENTS

This work has been partially supported by the Comisión Interministerial de Ciencia y Tecnología (CICYT), under the project "Navegación basada en visión de robots autónomos en entornos no estructurados" (TAP97-1209).

REFERENCES

[1] Brooks R. A. "A robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation, RA-2(1), pag. 14-26, 1986.

[2] Brooks R. A. "Intelligence without representation", Artificial Intelligence 47:139-159, 1991.

[3] Brooks R. A. "Intelligence without reason", MIT AI Memo 1293, 1991.

[4] Caruana R. "Multitask Learning: A Knowledge-Based Source of Inductive Bias", In Proceedings of the Tenth Int. Conf. on Machine Learning, 1993.

[5] Connell J. H. "A behavior based arm controller", IEEE Transaction on Robotics and Autonomous. Vol 5, N. 6, 1989.

[6] Dorigo, M., and Colombetti M. "Robot Shaping: developing autonomous agents through learning", Artificial Intelligence 71:321-370, 1994.

[7] Goldberg D. E. "Genetic Algorithms in search, optimization and machine learning", Addison Wesley, 1989.

[8] Kaelbling L. P., Littman M. L. and Moore A. W. "An Introduction to Reinforcement Learning", In "The Biology and Technology of Intelligent Autonomous Agents", Springer-Verlag, 1995.

[9] Koza J. R. "Genetic programming: On the programming of Computers by Means of Natural Selection" Cambridge, MA: MIT Press, 1992.

[10] Maes P. and Brooks R. A. "Learning to Coordinate Behaviors", Proceedings of the AAAI, 1990.

[11] Mahadevan, S. and Connell, J. "Automatic programming of behavior-based robots using reinforcement learning", Artificial Intelligence 55:311-365, 1992.

[12] Mataric M. J. "Navigation with a rat brain: A neurobiologically-inspired model for robot spatial representation", Proceedings of the First Int. Conf. on Simulation and Adaptive Behavior, MIT Press, 1990.

[13] Pfeifer R. "Cognition - Perspectives from Autonomous Agents", In "The Biology and Technology of Intelligent Autonomous Agents", Springer-Verlag, 1995.

[14] Sutton, R. S. and Barto, A. G. "Reinforcement Learning: An Introduction", A Bradford Book, The MIT Press, 1998.

[15] Utgoff, P. E. and Cohen, P. R. "Applicability of Reinforcement Learning", In The Methodology of Applying Machine Learning: Problem Definition, Task Decomposition and Technique Selection Workshop, ICML-98, pag. 37-43, 1998.

[16] Watkins C. J. C. H., Dayan P. "Q-Learning" Machine Learning, 8:279-292, 1992.