# Learning in Categorizable Environments

**Josep M Porta** **Enric Celaya**
Institut de Robòtica i Informàtica Industrial
Gran Capità 2-4, Edifici NEXUS, Planta 2, 08034, Barcelona
Phone: +34 93 401 57 91 Fax: +34 93 401 57 50
E-mail: {jporta,celaya}@iri.upc.es

## Abstract

Reinforcement learning constitutes a promising approach to achieve the adaptation of an animat's behavior to its environment. However, existing algorithms tend to be too time and space consuming, and become useless when the number of sensors and actions reach the typical sizes of a realistic animat situation. Existing generalization techniques mitigate somehow the problem, but they are still insufficient. In this paper, we suggest that the efficiency of learning observed in animals is possible, in part, because they capture important regularities, existing in the dynamics of its interaction with the environment, that are not exploited enough by the generalization algorithms. We characterize a kind of regularity that is plausible to hold for many animat environments, that we call *categorizability*, and propose an algorithm that takes advantage of such regularity. The main features of the algorithm are: a competitive evaluation of actions through different partial views of the same situation, and the on-line generation of new partial views to improve the accuracy of action evaluation. The described algorithm is applied to the task of learning to walk with a simulated six-legged robot.

## 1. Introduction

Probably, anyone working in automatic learning has been puzzled by the extraordinary ability exhibited by animals to learn. We are still far from understanding all the important aspects that contribute to their incomparable performance but, no doubt, one of the factors of their success is that they use restricted forms of learning. Animals are not able to learn arbitrary things in arbitrary conditions. It is well known, for example, that some species are able to learn to associate certain rewards with the percepts of certain sensorial modalities, but not with others. It is also clear that no animal can learn arbitrarily complex mappings or correlations between inputs and rewards: there are limitations in what can be possibly learned by each leaving being. From a computational point of view, we could say that their learning abilities are not general, but tailored to their available resources and specific needs for their survival.

In contrast, generality has been for longtime a requirement for most approaches to automatic learning. This generality is not without a cost. It gives rise to combinatorial explosion that makes learning problems become intractable as soon as their size approaches a realistic animat situation. This can not be avoided if we have to face completely arbitrary problems, as is the case in many synthetic learning situations. Hopefully, the interaction of an animat with its environment will present many regularities that can be capitalized by a learning agent, so that specific learning approaches can be devised that are much more efficient for these particular situations than a completely general one. The problem remains to identify which regularities can be assumed in a learning situation and how can they be taken into account by a specific learning algorithm.

In the context of Reinforcement Learning (RL) (Kaelbling et al., 1996), the regularity imposed to the environment is kept to a minimum, assuming that it can be characterized as a Markov process with a finite number of possible states and constant transition probabilities between them for each possible action executed by the animat. Often, the animat is not able to differentiate between all possible states with its sensorial apparatus, so that the environment, as experienced by the animat, turns out to be non-Markovian and must be formalized as a Partially Observable Markovian Decision Process (POMDP), what makes things even worse.

In non-toy environments, that is, environments with a large number of different states, the assumption of a (partially observable) Markovian environment is not enough to allow learning in acceptable amounts of time. The reason is that, even in the strictly Markovian case, the animat can not guess the result of an action in a given state before it has been experienced a number of times, and therefore, in general, it is necessary to explore a huge number of situations.

For efficient learning to be possible, in addition to the Markovian property, the environment must present some kind of structure, so that the effect of an ac-

tion can be predicted well enough by attending to only a small fraction of all the information contained in the whole state. The process of interpreting the world in terms of a reduced number of observable features constitutes a *categorization* of it. Plain Reinforcement Learning algorithms, as Q-Learning, ignore completely the problem of categorization, that is left as a designer's responsibility. Indeed, different extensions of Reinforcement Learning have been proposed to automatically generalize between states by clustering techniques (Mahadevan and Connell, 1992) or by successive subdivisions of the state space (McCallum, 1996), (Chapman and Kaelbling, 1991), so that states that are equivalent for the purposes of the learner are treated as a single state, thus simplifying the learning task. Formally this corresponds to a redefinition of the states of the Markovian process, so that its number is reduced substituting them for classes from which similar reinforcement can be expected.

We think that further improvements in the efficiency of learning can hardly be achieved by designing still more clever generalization algorithms, but what is needed, instead, even at the risk of losing generality, is to make stronger assumptions on the structure of the environment so that we can use more specialized algorithms that take advantage of such a structure. It seems that there is still plenty of room to make plausible assumptions on the environment in addition to the Markovian property and the feasibility of using simplified descriptions or clusterings of states.

We observe that, in many realistic situations, most of the sensor inputs are just irrelevant to predict the effect of a particular action in a given situation, that is, that the effect of a given action does not depend on the whole state of the world, nor on the full sensory information available to the animat, nor even on all the features that are actually relevant in other situations. Thus, for example, the value resulting from the action of "grasping the object in front of me" will depend on the object being food, the tail of a wild animal, or an unimportant object. However, the result will probably be the same no matter if it is day or night, if I am standing or lying down, or if I can see a red light nearby or not (aspects, all of them, that may become important in other circumstances).

This observation leads us to enunciate what we call the *categorizability hypothesis*, that is plausible for many animats, based on which we have developed a reinforcement learning algorithm whose main difference with previous approaches is that the result of each action is not associated with each state, but with selected partial views of the world. As we will see, when the categorizability hypothesis is fulfilled, the method becomes much more efficient in time and space than the usual, state-based, reinforcement learning algorithms. The rest of this paper is organized as follows: In the next section, the categorizability hypothesis is introduced and an approach to reinforcement learning based on what we call *partial views*, alternative to the usual state-based reinforcement learning, is analyzed in the context of the categorizability hypothesis. In Section 3, a learning algorithm based on partial views is presented, and it is applied to the task of learning to walk with a six-legged robot in Section 4. In Section 5, we compare our approach with other related works, and in Section 6, we present some conclusions and point to some directions in which this work could be extended.

## 2. Categorizability of Environments

We assume that the learning agent perceives the world through a set of binary *feature detectors* $f_i, i = 1 \ldots n$ triggered by stimuli received from its interaction with the environment. We say that the agent perceives a *partial view of order* $m$ $v(f_{i_1}, ..., f_{i_m}), m \leq n$ whenever features $f_{i_1}, ..., f_{i_m}$ are all simultaneously active. Note that many partial views can be perceived at the same time, for example, if $f_1$, $f_2$, and $f_3$ are active, then the agent perceives the partial view $v(f_1, f_2, f_3)$, but also $v(f_1), v(f_2), v(f_3), v(f_1, f_2), v(f_1, f_3)$, and $v(f_2, f_3)$.

If, in a given situation, some of the features are irrelevant to predict the effect of a given action, this means that the perception of the partial view defined by the remaining features (the relevant ones) is enough to make the prediction. Our hypothesis that most features will be often irrelevant for predicting the effect of a given action implies that predictions can be made using only low order partial views. Then, we say that the environment is *categorizable* with respect to a learning task, if the reward obtained from the execution of each action depends only on low order partial views (that may be different for each action).

Note that categorizability is not a binary property but a graded one. An extreme case, in which we would say that the environment is completely categorizable, would be one in which the result of each action could be predicted from a number of (mutually exclusive) partial views of order one. On the other extreme, if the effect of each action could only be predicted by taking into account the value of all feature detectors, we would say that the environment is not categorizable at all. The *categorizability hypothesis* states that the actual situation is much closer to the first case than to the second.

### *2.1 Action evaluation with partial views*

As far as we can tell, at the core of all existing reinforcement learning algorithms is that the *state* of the system must be used to predict the expected reward of *all* possible actions. Existing generalization techniques are based on the following consideration: If two states coincide in their reward predictions for all actions, then these two

states do not need to be differentiated and can be considered as a single state. Clustering techniques, as well as feature-based approaches, simply provide different ways to joint or split states based on this equivalence principle. This implies that, as soon as two states differ in their reward predictions for just one action, they must be kept as different states, no matter how coincident they are in the predictions for the remaining actions, so that many opportunities of useful generalization are lost. Associating the expected reward of an action to a partial view permits such generalization. The drawback, in the general case, is that the number of possible partial views is much larger than the number of states. However, *if the environment is categorizable, we will need to consider only the partial views of lower order, and it will be advantageous to store the expected reward of an action for each partial view instead of for each state.*

We define the value $q_v(a)$ associated with action $a$ and partial view $v$ as the mean discounted accumulated reward received after executing action $a$ while $v$ is observed. This is an averaged value that provides no information about the distribution of values taken around this mean value in different situations. Distributions with low dispersion are indicative of coherent reward, or as we say it, *high relevance* of the partial view for this action, meaning that $q_v(a)$ is a very accurate prediction of the reward that will be obtained. Non-relevant partial views with high dispersion value distributions provide little reliable estimations of the future reward. Thus, we will maintain an estimation of the relevance of each partial view for each action, and use the relevance in addition to the mean value of each partial view to determine the expected reward of each action in a given situation.

The estimation of $q_v(a)$ is in many aspects similar to that of the value of an action in a state of Q-learning, but there is an important difference that prevents the direct use of the Q-learning algorithm as is. The difference is that while in Q-learning only one state is observed at a time and the value of each action can be determined from it, in our approach, many partial views can be simultaneously observed, each one providing a different evaluation for the same action. The problem is then, how should the value be obtained? Two possible solutions come to mind: using a weighted sum of the values predicted by all the observed partial views, or using a competitive approach, in which only the most relevant partial view for each given action is used to determine the predicted value. As we explain next, both solutions may produce better or worst results depending on the relationship existing between partial views.

We say that *two partial views are redundant* if they respond to alternative views of the same source of reward. Redundant partial views are predicting essentially the same and, though they can increase the confidence on the prediction, it would be misleading to add up both value predictions. Conversely, we say that *two partial views are independent* if they predict rewards associated with different aspects of the situation, each corresponding to a different source of reward, so that their value predictions should be added. An example will serve to clarify this: Consider the action of reaching for an object and two features of the object: smell and temperature. They define independent partial views: warm objects provide warmth and smelly food provides aliment, and both are independent. We should evaluate the action with the sum of the values predicted by each partial view. However, if we had also the possibility to taste the food, it only would confirm the information provided by smell, making the reward prediction more certain, but it cannot be added to the reward already well predicted by smell only.

If we could know if two partial views are redundant or independent, we could use the appropriate composition rule for reward prediction, but the problem is that we have no way, in principle, to guess it. In the lack of the necessary information, we must adopt one of the composition rules: weighted sum, as if partial views were always independent, or winner-takes-all, as if they were always redundant. In both cases, it is possible to compensate the effect of unwanted-type partial views: In the weighted sum solution, the over-evaluation arising from redundant partial views could, in principle, be compensated for by introducing an additional partial view corresponding to the simultaneous observation of the redundant ones and providing opposite compensatory reward, so that if two redundant partial views were observed, the result would be the same as if only one of them was observed. With the winner-takes-all solution, the problem arises when the contributions of several independent partial views must be added up. This can be solved in a similar way, by introducing a new partial view corresponding to the simultaneous observation of the independent ones and providing the accumulated prediction. In this case, since the new partial view is more specific (a conjunction of the independent ones), it will become more relevant than both of them whenever the two independent partial views are active, and will win the competition to give the correct result.

However, there is an advantage of the competitive solution over the weighted sum one. Since redundant partial views provide robustness to the system and excessive redundancy degrades efficiency, we may want to create or remove redundant partial views at our convenience. Then, when the competitive solution is used, adding or removing redundant partial views has no effect on the evaluations. But, if a weighted sum is used, the distributed nature of the evaluation means that the removal of a partial view must be followed by a consistent modification for many partial views, making the system maintenance more complex and unstable. This is the main

**Learning Algorithm**
*(Initialize)*
$F \leftarrow$ Set of features detectors
$PV \leftarrow \{v(f) | f \in F\}$
**For each** $v$ **in** $PV$
**For each action** $a$
$q_v(a) \leftarrow 0$
$e_v(a) \leftarrow 0$
$i_v(a) \leftarrow 0$
**endfor**
**endfor**
$q_{max} \leftarrow -\infty$
$q_{min} \leftarrow +\infty$
$V \leftarrow \{v \in PV | v \text{ is observed}\}$
**Do forever:**
$a \leftarrow$ Action Selection
Execute $a$
*(System Update)*
$r_a \leftarrow$ *Reward* generated by $a$
$V_{ant} \leftarrow V$
$V \leftarrow \{v \in PV | v \text{ is observed}\}$
Statistics Update
Partial View Management
**enddo**

Algorithm 1: Top level sketch of the learning algorithm.

reason why we choose the competitive, winner-takes-all solution for the evaluation of action values.

## 3. The Learning Algorithm

We assume that the animat is provided with a set of binary feature detectors[1] and a set of actions. The feature detectors are used to define an initial set of partial views ($PV$) including all the partial views of order 1. After the initialization phase, the algorithm enters in a continuous loop consisting in choosing an action to be executed (using the Action Selection procedure detailed in section 3.1), executing it, and updating the system so that its performance improves in the future. The system update includes the statistics update (see section 3.2) and the partial view management (section 3.3).

We store three values for each partial view $v$ and action $a$:

- Reward estimation: $q_v(a)$.
- Error estimation: $e_v(a)$.
- Confidence index: $i_v(a)$.

[1] Feature detectors ($f$) with multiple values $\{b_1, \ldots, b_k\}$ can be binarized using $k$ predicates of the form $(f = b_i)$.

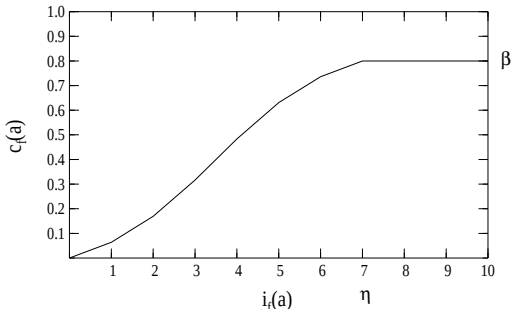


Figure 1: Confidence function with $\eta$=7 and $\beta$=0.8.

The reward estimation $q_v(a)$ approximates the average discounted reward derived from the execution of action $a$ when partial view $v$ is perceived. The error estimation $e_v(a)$ approximates the average absolute value of the error in the previous statistics. To estimate the confidence in the reward and error statistics we use a confidence index $i_v(a)$ that keeps track of the number of times action $a$ has been executed after observing the partial view $v$. The confidence is derived from $i_v(a)$ using a *confidence_function* in the following way:

$$c_v(a) = \min\{\beta,\ \textit{confidence\_function}(i_v(a))\},$$

where the *confidence_function* is a strictly increasing function, and $\beta$ is a top value for the confidence. The confidence is used to weight the influence of new collected data on the reward and error statistics (the lower the confidence, the higher the weight). In general, $\beta$ is less than 1 so that the statistics are always under revision. Different confidence schemas can be implemented in our system by changing the *confidence_function*. We use a sigmoid-like confidence function (see figure 1) that increases slowly for low values of $i_v(a)$ reducing the importance of the first obtained rewards which, in general, are very noisy. A parameter ($\eta$) determines the point at which this function reaches the top value $\beta$.

Finally, $q_{min}$ and $q_{max}$ are the extreme values of $q_v(a)$ effectively observed up to now. These bounds are updated with the procedure described in section 3.2, and are used in the action selection procedure (section 3.1) to estimate the possible effects of an action when no reliable information is available.

### *3.1 Action Selection*

As mentioned above, to evaluate an action in a given situation, instead of combining the reward predictions provided by each observed partial view, we use a competitive approach. Two factors are taken into account to select the winner view for each action (i.e., the view that provides the best guess for reward prediction):

- The *relevance* $\rho_v(a)$: The lower the average error, the more reliable the reward prediction. So, we define the *relevance* ($\rho_v(a)$) of the partial view $v$ with respect

**Action Selection**
  **For each action** $a$
    $v \leftarrow Winner(V, a)$
    $guess(a) \leftarrow q_v(a) + random(-2e_v(a), 2e_v(a))$
    $noise \leftarrow random(q_{min}, q_{max})$
    $guess(a) \leftarrow c_v(a)\ guess_a + (1 - c_v(a))\ noise$
  **endfor**
  **return**$(\arg \max_{\forall a}\{guess(a)\})$

Algorithm 2: Action Selection procedure. The $random(a, b)$ function generates a random number in the interval $[a, b]$.

to action $a$ mapping the error measure in the interval $[0, 1]$:

$$\rho_v(a) = 1/(1 + e_v(a))$$

- The *confidence* $c_v(a)$: The higher the confidence on a reward prediction, the more reliable the prediction.

The winner view is the one that maximizes the product of these two factors:

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

The winner view $v$ for each action $a$ is used to get a *guess* of the effects of that action. This guess results from the combination of the corresponding reward and the error estimations ($q_v(a)$ and $e_v(a)$, respectively). Assuming, for simplicity, a uniform distribution, the reward derived from action $a$ when $v$ is perceived can be any value inside the interval

$$I_v(a) = [q_v(a) - 2e_v(a),\ q_v(a) + 2e_v(a)]$$

with the same probability.

This guess is altered according to the confidence (i.e., certainty) of the reward and error statistics used to derive it: if the animat does not know much about an action, its effects could be quite different from the predicted ones. For this reason, the guess for each action is modified with a noise signal weighted by the confidence (the lowest the confidence, the higher the effect of the noise). This noise favors exploration of the non well-tested actions and automatically adapts the exploration rate.

Finally, the action to be executed is simply the one that produced the highest guess.

### *3.2 Statistics Update*

The system update uses the effects of the last executed action ($a$) to update of the statistics of all observed partial views.

**Statistics Update**
  $q \leftarrow r_a + \gamma \max_{\forall a'}\{q_v(a') | v = Winner(V, a')\}$
  $q_{min} \leftarrow \min\{q_{min}, q\}$
  $q_{max} \leftarrow \max\{q_{max}, q\}$
  **For each** $v$ **in** $V_{ant}$
    $q_v(a) \leftarrow c_v(a)q_v(a) + (1 - c_v(a))q$
    $e_v(a) \leftarrow c_v(a)e_v(a) + (1 - c_v(a))|q_v(a) - q|$
    **if** $q \in I_v(a)$ **then**
      **if** $c_v(a) < \beta$ **then**
        $i_v(a) \leftarrow i_v(a) + 1$
      **endif**
    **else**
      $i_v(a) \leftarrow i_v(a) - 1$
    **endif**
  **endfor**

Algorithm 3: Statistics Update procedure.

The effects of an action $a$ when a given partial view $v$ is observed can be defined (using a Bellman-like equation) as

$$q^*_v(a) = \overline{r}_a + \gamma \sum_{\forall V} p(v, a, V)\ q^*(V),$$

where $\overline{r}_a$ is the average reward obtained immediately after executing $a$, $\gamma$ is the discount factor used to balance the importance of immediate with respect to delayed reward, $q^*(V)$ represents the goodness of the situation $V$, and $p(v, a, V)$ is the probability of reaching that situation after the execution of $a$ when $v$ is perceived. The goodness of a situation is assessed using the best action executable in that situation

$$q^*(V) = \max_{\forall a'}\{q^*_v(a') | v = Winner(V, a')\}$$

since this gives us information about how well the animat can perform (at most) from that situation.

In the statistics update procedure (algorithm 3), $q_v(a)$ is adjusted for all observed partial views, using a temporal difference rule, so that it progressively approximates $q^*_v(a)$. At the same time, $e_v(a)$ is adjusted using a identical rule, so that it estimates the average absolute value of the error of $q_v(a)$. Observe that both statistics are updated using the confidence as a learning rate, which initially is 0, and consequently, the initial values of $q_v(a)$ and $e_v(a)$ have no influence on the future values of these variables. These initial values become relevant when using a constant learning rate, as many existing RL algorithms do.

If the observed effects of the last executed action agree with the current estimate interval for the reward ($I_v(a)$), then the confidence index is increased by one unit until the value $\beta$ is reached. Otherwise, the confidence index is decreased allowing a faster adaptation of the statistics to the last obtained, surprising values of reward.

**Partial View Management**
*(Partial View Elimination)*
$PV \leftarrow PV - \{v \in PV \mid redundancy(v) > \lambda\}$
$v \leftarrow Winner(V_{ant}, a)$
**if** $|q_v(a) - q| > \delta$ **and** $\|PV\| = \mu$ **then**
$WV \leftarrow$ The $\tau$ partial views from $PV$ with:
- Lowest $creation\_error(v)$, and
- $creation\_error(v) < |q_v(a) - q|$
$PV \leftarrow PV - WV$
**endif**
*(Partial View Generation)*
**if** $|q_v(a) - q| > \delta$ **then**
$t \leftarrow 0$
**while** $\|PV\| < \mu$ **and** $t < \tau$
Select two different views $v_1$, $v_2$ from $V_{ant}$
preferring those that minimize
$c\,|q_{v_i}(a) - q| + (1 - c)(q_{max} - q_{min})$
with $c \leftarrow c_{v_i}(a)$
Create a new view $v' = (v_1 \oplus v_2)$
$creation\_error(v') \leftarrow |q_v(a) - q|$
$PV \leftarrow PV \cup \{v'\}$
$t \leftarrow t + 1$
**endwhile**
**endif**

Algorithm 4: Partial View Management procedure. The value of $q$ is calculated in the Statistics Update procedure.

### *3.3 Partial View Management*

This procedure (Algorithm 4) includes the generation of new partial views and the removal of previously generated ones that proved to be useless.

A situation needing better categorization than that provided by the current set of partial views can be identified because actions executed in that situation have an effect ($q$) too different from the expected one ($q_v(a)$). The larger the difference between the predicted and the observed values, the more critical is to improve the categorization of this situation. In our implementation, new partial views are generated when this difference exceeds a given threshold ($\delta$). $\delta$ is determined so that the intrinsic noise in the reward signal does not produce the generation of new partial views.

Every time a wrong prediction is made, at most $\tau$ new partial views are generated by combination of pairs of observed partial views. The combination of two partials views $v_1 \oplus v_2$ consists in a new partial view that includes all the features included in either $v_1$ or $v_2$. Observe that the combination of two different partials views is of a higher order than its components, and it will be observed in those situations in which both components are simultaneously observed.

We favor the combination of views with high confidence, so that the combinations of the recently created ones are discouraged. Additionally, we bias our system to favor the combination of those views ($v_i$) whose reward prediction ($q_{v_i}(a)$) is closer to the observed one ($q$), since they are more likely to be active in situations similar to the one we are trying to categorize. Finally, the generation of views syntactically equivalent to already existing ones is not allowed.

Observe that, if necessary, a specific view can be generated for each different situation, in which case our algorithm would end up working like a usual state-based RL algorithm. However, according to the categorizability hypothesis, this extreme possibility can be discarted. For this reason, to improve efficiency, we limit the number of partial views to a maximum of $\mu$. To keep this limit, we must eliminate the less useful partial views.

A partial view $v = v_1 \oplus v_2$ can be removed if its reward predictions are too similar to those of the component views $v_1$, $v_2$.

The similarity between two reward distributions, that we have assumed to be uniform, can be computed as the normalized degree of intersection between the corresponding reward intervals:

$$similarity(I_v(a), I_{v'}(a')) = \frac{\|I_v(a) \cap I_{v'}(a')\|}{\max\{\|I_v(a)\|, \|I_{v'}(a')\|\}}.$$

Based on this similarity measure, we define the *redundancy* of a partial view $v = (v_1 \oplus v_2)$ as:

$$redundancy(v) = \min_{\forall a}\{\max\{similarity(I_v(a), I_{v_1}(a)), similarity(I_v(a), I_{v_2}(a))\}\}.$$

Partial views with a redundancy above a given threshold ($\lambda$) are eliminated. Since the redundancy of a partial view can only be estimated after observing it a number of times, the redundancy of the partial views with low confidence index is set to 0 so that they are not removed.

Additionally, if it is necessary to generate new views but the maximum number of views has been reached, those partial views with lower creation error than the current one are eliminated.

## 4. A Test Example: Gait Generation for a Six-Legged Robot

### *4.1 Experiment Setup*

We applied our algorithm to the task of learning to walk with a simulated six-legged robot (figure 2). The simulator allows to command each leg of the robot in two independent degrees of freedom (horizontal and vertical) and provides a mechanism that automatically compensate the forward movement of stepping legs by moving

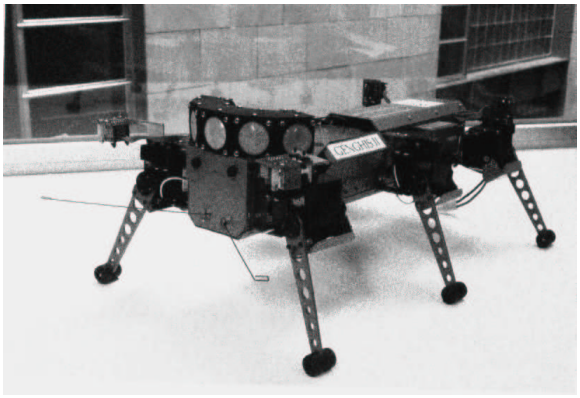

Figure 2: The Genghis II walking robot.

backward all the legs on ground (the *alpha-balance* behavior described in (Brooks, 1989)). With this mechanism, the task to be learnt consists in deciding at every moment which legs must step (that is, leave the ground and move to an advanced position), and which must descend (stay on the ground to support and propel the body). The sequence of executed steps is known as the *gait* of the robot. The simulator is able to detect when the robot is in an unstable position.

We defined a set of 24 feature detectors:

- **On_the_air(x):** Active if the leg $x$ is on the air.
- **On_the_ground(x):** Active when the *On_the_air(x)* feature is not active.
- **Advanced(x):** Active if leg $x$ is ahead the reference position.
- **Behind(x):** Active when the *Advanced(x)* feature is not active.

With this set of features the number of possible different situations is 4096.

The set of actions consists of all the combinations of the two possible actions available to each leg. This includes from touching the ground with all legs to stepping all legs simultaneously.

The reward signal includes two aspects:

- **Stability:** If an action will cause the robot to fall down, a reward of -50 is given and the action is not executed, so that, actually the robot never falls down.
- **Efficiency:** A reward equal to the distance advanced by the robot is given.

It is well known that the most efficient stable gait is the *tripod gait* (Wilson, 1966) in which two sets of three legs (leg sets $(L1, R2, L3)$ and $(R1, L2, R3)$ in figure 3) step alternatively. Using this gait, the robot would obtain a

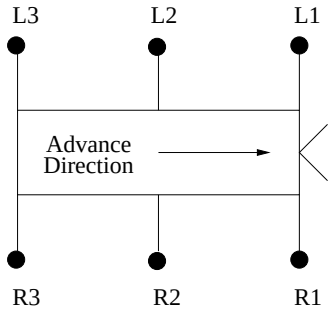


Figure 3: Diagram of the six-legged robot.

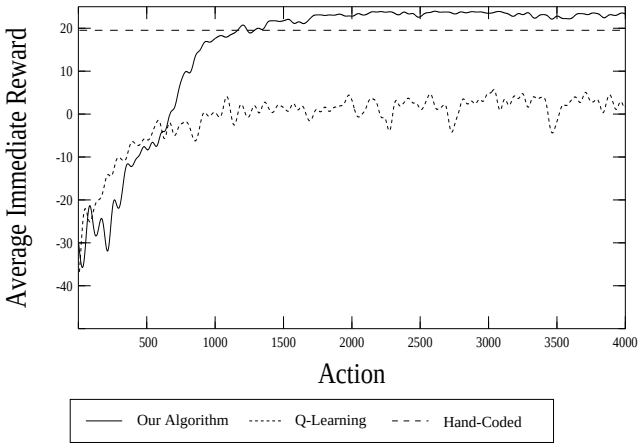


Figure 4: Performance of our algorithm compared with Q-Learning and a hand-coded behavior.

reward of 50 (when one group of three legs steps) followed by a reward of 0 (when those legs recover contact with ground), so that the optimal average reward is 25.

In the experiments, the robot is set in an initial posture with all the legs in contact with the ground but in a random advance position. For these experiments, we use the following set of parameters: $\gamma = 0.9$, $\beta = 0.99$, $\eta = 10$, $\delta = 10$, $\tau = 3$, $\mu = 124$ and, $\lambda = 0.95$.

## *4.2 Results*

Figure 4 shows the results obtained with our learning algorithm compared with those obtained using standard Q-Learning (Watkins and Dayan, 1992). For this algorithm we use a learning rate $\alpha = 0.1$ and an action selection that performs exploratory actions with probability 0.1. The states used by Q-Learning are the 4096 possible situations identifiable by the 24 feature detectors described before, and the set of actions is the same as the one used in our algorithm. As a reference, we show the results using a hand-coded gait generation rule previously developed by ourselves (Celaya and Porta, 1998).

Figure 4 shows that our algorithm converges to a so-

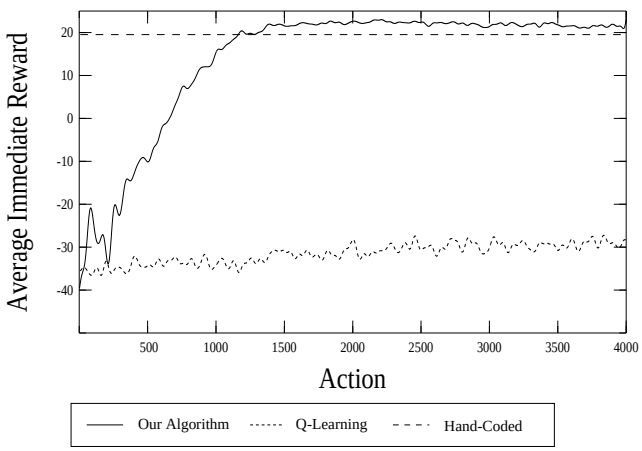


Figure 5: Performance of our algorithm compared with Q-Learning when there are irrelevant features.

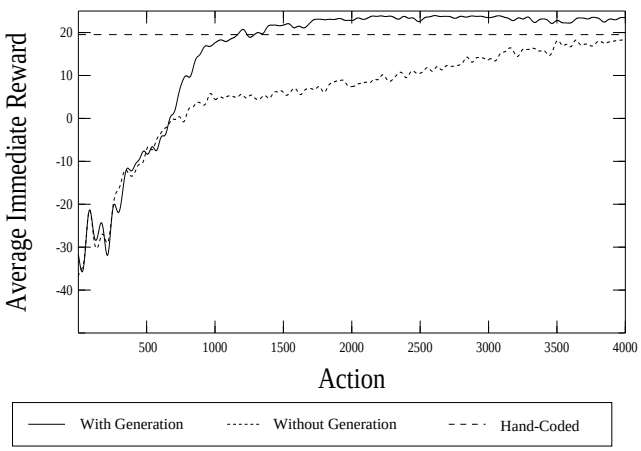


Figure 6: The performance with and without the partial view generation procedure.

lution much more close to the optimal value (25) than Q-Learning (that gets an average immediate reward close to 0) and its performance is much more stable. Moreover, if we consider the amount of space used by each algorithm we find that Q-Learning stores one value for each state and action ($4096 \cdot 64 = 262144$) and our algorithm stores three values for each partial view and action ($3 \cdot 124 \cdot 64 = 23808$). This supposes that our algorithm uses less than 10% of the memory used by Q-Learning.

Observe that the performance attained using our algorithm is even superior to that of the hand-coded strategy. This is because there is a restricted set of initial postures (that can appear when choosing initial postures at random) from which our rule does not perform optimally (Porta and Celaya, 1998).

The advantage of our algorithm with respect to state-based ones is increased in problems in which some of the sensors provide information not related with the task. To test this point, we set up an experiment in which six feature detectors (that get active randomly) were added to the 24 initial ones. With these new features, the number of possible combinations of feature activations increases, and so does the number of states considered by Q-Learning. Figure 5 shows the comparison between our algorithm and Q-Learning for this problem. Q-Learning is not able to learn a reasonable gait strategy, while the performance of our algorithm is equivalent to the one obtained without irrelevant features. This means that our algorithm is able to detect which sets of features are relevant and use them to determine the robot's behavior. It is remarkable that, in this case, the ratio of memory used by our algorithm with respect to that used by state-based algorithms, drops below 0.15%. This exemplifies how the performance of the state-based algorithms degrades as the number of features increases while this is not necessarily the case using the partial view approach.

The importance of the generation of partial views in the improvement of the categorization can be seen comparing the results obtained for the same problem with and without this mechanism (figure 6). While the results show that the task can be learnt acceptably well with partial views of order 1, the learning is faster when higher order views are generated, and the final performance is improved in a significant way. The generated partial views are specially useful for the efficiency aspect of the gait generation problem. In general, the reward obtained when executing a given action is higher if the stepping legs are *behind* than if they are *advanced*. If more than one leg is to be stepped simultaneously (as for instance happens in the tripod gait), then the *behind* and *advanced* features of these legs should be combined to identify whether or not it is appropriate to execute the corresponding action. For instance, a partial view generated by our system in a typical run is:

$$v(Advanced(L1), Advanced(R2), Advanced(L3))$$

This feature identifies situations in which the three legs of one of the tripods are simultaneously advanced (probably, because they have stepped recently). Of course, in this situation, stepping these three legs does not produce a great advance of the robot. The observation of the above partial view predicts a low reward for the action that makes a step with legs of one of the tripod (L1, R2 or L3) and a high one for the action that makes a step with the legs of the other tripod.

## 5. Related Work

The idea of using feature detectors as a base of the RL algorithms is not new (see (Boutilier et al., 1999) for a survey). In general, previous work on feature-based RL aims at finding state definitions from features detectors. A decision tree is used to classify each situation accord-

ing to some of the features active in that situation. The information stored in each leaf of the decision tree is used to predict the effects of all actions for the situations that are classified in that leaf.

In (Chapman and Kaelbling, 1991), the decision tree is constructed assuming that each feature detector is individually relevant and, consequently, this approach has problems when a given situation has to be categorized paying attention to more than one feature simultaneously. In our approach, the partial view composition procedure avoids this inconvenience. By its side, (McCallum, 1996) tries to find features that make utile distinctions between states with the result that situations that are equivalent with respect to a given action must be classified in different leafs of the tree if they are not equivalent with respect to other actions. In our partial view approach, the relevance is independently estimated for each view and action, and a given partial view only include features that are relevant for a given action.

It is remarkable that McCallum includes the use of features active in previous time slices to build the decision tree so that non-Markovian problems can be tackled. A similar technique could be included in our algorithm so that it could also deal with this kind of problems.

The system described in (Wilson, 1995) is similar to our partial view approach, since it also aims at determining the relevance of *classifiers* (that are combinations of features and an associated action), without referring to a global state. Moreover, Wilson also proposes a measure of relevance of the classifiers based on the error of the reward prediction. The main difference between this approach and ours is that Wilson's work uses a genetic algorithm to find the relevant classifiers, while we advocate for the use of an incremental strategy in which the rate at which new partial views are created depends on the situations the animat encounters and not on the dynamics of a search mechanism (the genetic algorithm) that works independently of the animat's experiences. In this way we do not lose any chance of learning and the animat behavior can adapt to new situations more efficiently. Furthermore, Wilson's approach uses a weighted sum of the predictions of the classifier advocating for each action to determine the expected effect of that action, while we propose to use a winner-takes-all strategy.

Another approach related to ours is described in (Maes and Brooks, 1990). They also confront the problem of step coordination with a six-legged robot and try to find a precondition for the execution of each action in the appropriate moment. However, their method is restricted to problems with immediate reward only, while our algorithm is able to deal with delayed reward. Furthermore, they limit the precondition to be a conjunctive predicate of binary perceptual conditions (i.e., binary feature detectors), while we allow disjunctions of conjunctive predicates (many partial views can be relevant for a given action, and each partial view can be seen as a conjunctive predicate).

## 6. Conclusion and Future Work

The approach taken in this paper owes a lot to a reflection about how biological animals learn and behave. It is difficult to think that animal behavior is produced by means of the permanently running cycle typical of reinforcement learning: observe the current state, compute and execute the most promising action for the current state, get the resulting reward, update the policy. On the contrary, animals seem to follow a "minimum effort policy" whenever possible, and try to avoid acting when this is not necessary. The behavior of animals is perhaps best explained as a sequence of reactions to relevant stimuli (either caused by external inputs or by internal motivations), instead of as a constant involvement about the problem of "what to do next", which appears to be an artifact problem produced by the way we design our animats, in which the output of the control process must be explicitly specified at any time.

These thoughts yielded us to a subtly different view about which should be the goal of learning: While in Reinforcement Learning it is assumed that the agent must learn a policy, i.e., a mapping from states to actions, we propose learning "to what stimuli is it worth to react with a given action". There are situations for which acting is not clearly correlated with reward, and it is probably not worth the effort to learn the exact outcome of each possible action in such situations. It is much more useful to concentrate the efforts in discovering those situations in which the execution of a given action really matters. This is, in a rather informal sense, what our partial view-based algorithm tries to do.

In this paper we have introduced the notion of environment categorizability. It is not possible to know a priori (except for trivial cases) whether or not an environment is categorizable by a given animat. The state-based RL implicitly assumes that all possible combination of features have to be taken into account separately. Our approach assumes just the opposite: that the environment is categorizable and consequently only reduced combinations of features need to be taken into account. The drawback of using the state-based approach is that realistic problems become intractable since the number of possible combinations of features is exponential in the number of features. This is known as the *curse of dimensionality*. Since our approach limits the use of combinations of features, it is not affected by this problem. Of course, our approach is less advantageous as the environment is less categorizable. However, the preliminary results obtained with our algorithm suggest that in realistic situations our hypothesis is valid and consequently our approach can work much more efficiently than existing ones in many cases.

The particular mechanisms used in the presented algorithm could be extended in several ways. Aspects that deserve special consideration are:

- Confidence estimation: Given that the action selection, statistics update, and partial view generation procedures are dependent on a correct estimation of the confidence, any improvement in this measure will have an important repercussion. It would be interesting to adjust the confidence in a more adaptive way: up to now, we modify the confidence in fixed steps, but it would be sensible to do it proportionally to the difference between the predicted and the actual reward.

- Partial view management: The partial view generation mechanism can be improved incorporating new heuristics to select the views to be combined. By its side, the partial view elimination mechanism could be based on preserving those views actually useful (not those potentially useful as we do now) allowing a reduction in the size of the set of partial views.

The work presented in this paper makes a step towards the adaptation of the RL framework to animats. In this paper we review the assumptions about the environment and how the animat perceives it, moving from a state-based perception of the environment to a feature-based one. However other aspects of the RL framework deserve also consideration (Porta and Celaya, 1999). For instance, RL supposes that the actions the animat can execute are mutually incompatible and, from the point of view of an animat with a complex motor apparatus that can execute parallel actions, this is not adequate. Additionally, RL assumes that the direct effects (or what is the same, the direct reward) of an action is perceived immediately after its execution but in real animats this is not always true: the effect of an action can be perceived by the animat many time steps after its execution. This poses a temporal credit assignment problem not usually considered in RL. New RL algorithms more suited to the animat problem are necessary to overcome the challenge of controlling increasingly complex animats in increasingly complex environments. These new control systems will improve our knowledge about how adaptive behavior is possible in complex environments.

## Acknowledgements

We would like to thank the useful comments of two anonymous referees that greatly contributed to improve the quality of this paper.

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

## References

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 11:1–94.

Brooks, R. A. (1989). A Robot that Walks: Emergent Behaviors from a Carefully Evoled Network. *Neural Computation*, 1(2):253–262.

Celaya, E. and Porta, J. M. (1998). A Control Structure for the Locomotion of a Legged Robot on Difficult Terrain. *IEEE Robotics and Automation Magazine, Special Issue on Walking Robots*, 5(2):43–51.

Chapman, D. and Kaelbling, L. P. (1991). Input Generalization in Delayed Reinforcement Learning: An Algorithm and Performance Comparisons. In *Proceedings of the International Joint Conference on Artifical Intelligence*.

Kaelbling, L. P., Littman, M., and Moore, A. W. (1996). An Introduction to Reinforcement Learning. *Journal of Artificial Inteligence Research*, 4:237–285.

Maes, P. and Brooks, R. A. (1990). Learning to Coordinate Behaviors. In *Proceedings of the AAAI-90*, pages 796–802.

Mahadevan, S. and Connell, J. (1992). Automatic Programming of Behavior-Based Robots Using Reinforcement Learning. *Artificial Intelligence*, 55:311–363.

McCallum, A. K. (1996). *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, Department of Computer Science, University of Rocherter, New York.

Porta, J. M. and Celaya, E. (1998). Gait Analysis for Six Legged Robots. Technical Report IRI-DT-9805, Institut de Robòtica i Informàtica Industrial.

Porta, J. M. and Celaya, E. (1999). Reinforcement Learning and Automatic Categorization. In *Proceedings of the 7th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 159–166.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Wilson, D. M. (1966). Insect Walking. *Annual Rev. pf Entomology*, pages 103–122.

Wilson, S. W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175.