Increamental Learning for Robot Control

I-Jen Chiang
chiang@robot.csie.ntu.edu.tw
Jane Yung-jen Hsu
yjhsu@csie.ntu.edu.tw
Department of Comuter Science and Information Engineering
National Taiwan University
Taipei, Taiwan 106, R.O.C.

Abstract

A robot can learn to act by trial and error in the world. A robot continues to obtain information about the environment from its sensors and to choose a suitable action to take. Having executed an action, the robot receives a reinforcement signal from the world indicating how well the action performed in that situation. The evaluation is used to adjust the robot's action selection policy for the given state. The process of learning the state-action function has been addressed by Watkins' Q-learning, Sutton's temporal-difference method, and Kaelbling's interval estimation method. One common problem with these reinforcement learning methods is that the convergence can be very slow due to the large state space. State clustering by least-square-error or Hamming distance, hierarchical learning architecture, and prioritized swapping can reduce the number of states, but a large portion of the space still has to be considered. This paper presents a new solution to this problem. A state is taken to be a combination of the robot's sensor status. Each sensor is viewed as an independent component. The importance of each sensor status relative to each action is computed based on the frequency of its occurrences. Not all sensors are needed for every action. For example, the forward sensors play the most important roles when the robot is moving forward.

1 Introduction

A robot can learn to act by trial and error in the world. The robot obtains the environment information form its sensors and chooses an action to take. Having executed an action, the robot will indicate how well the action was performing at that situation. From those state-action performing evaluation, the robot will gradually improve its action choosing policy for every states. For the robot control, all we need to do is to construct its state-action control function. It is not a easy work in the nonlinear domain. We need a refinement process to rectify the state-action [1] [2] to make it adaptive to the real world. The procedure of the state-action choosing policy refinement is a learning process. How to quickly and concisely create a state-action controller for the robot in dynamic environment by a learning process is the main purpose in this paper.

The reinforcement learning method [3] [4] [5] [6] [7] [8] [9] [10] [12] [13] [14] [15] has been addressed to construct the state-action function for the robot control, such as Watkins' Q-learning [15] [8] [9], Sutton's temporal-difference method [12] [13], and Kaelbling's interval estimation method [5] [6] [7]. One common problem with these reinforcement learning methods is that the convergence can be very slow due to the large state space. State clustering by least-square-error or Hamming distance [14], hierarchical learning architecture [8] [9], and prioritized swapping can reduce the number of states, but a large portion of the space still has to be considered.

This paper presents a new solution to this problem. A state is taken to be a combination of the robot sensor status. Each sensor is viewed as an independent component. The sensors on different states play different roles in performing an action. For a sensor-based robot, it is difficult to enumerate all the mappings of the states and the actions to construct the robot controller. In order to simplify all the state-action mappings, the learning process needs to identify which the robot sensor status play the roles of assistant, the resistant, or don't care of executing that action. As we seen, while a robot executing a forward action, the forward sensors of it play the most important roles to govern such operation. When the forward sensors sense that the robot is approximate to an obstacle, it is not difficult to understand that the robot needs to change the forward action; that is, those sensors status are the resistant to the forward action, but if they sense the robot is far from an obstacle it is easy to perform the forward action; that is, those sensors states are the assistants to the forward operation. If we can clearly identify the sensors' characteristics for each robot action, the consideration of sensor status for the action choosing policy will be simplified. The different roles of sensor status are represented as sensor status-action preference

0-7803-2559-1/95 $4.00 © 1995 IEEE

values. The sensor status-action preference values for all sensor status and actions will demonstrate their effects in the action-selection policy. The bigger the preference value of a sensor status to an action, the more important the sensor status to assist it. The value of the resistant is the minus sign value and the preference value of the don't care status is zero.

This paper presents an unsupervisory, incremental learning method to adapt robot operation in the environment. In our framework, there are some primitive actions given to the robot. The controller is a state-action mapping function, which utilizes the previous performing evaluation and the sensor status-action preference values of the state, to determine which action is the most adaptive. The learning process is based on the reinforcement learning methodology, which includes three main procedures: (1) observe the environment information from the robot sensors; (2) choose the most suitable action according to state-action performing evaluation; (3) cluster and adjust the values of the evaluation function and the sensor status-action preference function for each robot state and action.

At each moment in time, the robot gets information about the world form its sensors. According to those sensor status and the current sensor status-action preference values, we can sum up all the values for each action individually to determine which action is the most suitable, and perform that action. If the robot succeeds to perform that action, all the sensor status-action preference values will be increased by a reward; otherwise, all of them get the punishment. After the status-action preference refining process has proceeded, we make a normalization on it. If a sensor status has the role of assistant and resistant concurrently, plus these two different values to make it play only one role. It is called the normalization. The reward or the punishment given is according to the error rate and the learning time it takes. The lower the error, the smaller the reward and the punishment. The reward and the punishment will approach to constant values in a long run. Initially, the reward and the punishment are constant values. As by the simulated annealing procedure, they approach to zero.

Our method avoid the credit assignment problem. It is difficult to determine the values of the reward and the punishment in the reinforcement learning process. With unsuitable reward or punishment, the learning could bring to no effect. In our learning method the reward and punishment, which are initially constant, are decreased by the error and the learning duration, and applied to each sensor status individually at any moment. By the normalization process, we can quickly identify the sensor status to be the assistant, the resistant, or the don't care status.

In our experiments on mobile robot, we take 16 sonic sensors, which are independent one another around the robot, and divide their sensed range into four fixed status. And there are four primitive actions, such as moving forward, moving backward, turning right, turning left, given to the robot. We want to use those sensory information and those basic actions to construct a controller for mobile behavior, such as following the wall, moving in the corridor, avoiding the fixed and moving obstacle. As to the simulated mobile robot process on SPARC I, let the robot self-organize the behavior controller by our learning process in some different generated areas. The error rate can reduce to 1% within in five minutes at hundreds of steps.

Based on the reinforcement learning method to carry a conceptual learning [11] into effect, we can identify the feature of every sensor status for each action. By the normalization process, we can quickly distinguish what sensor status is no use for each action. When getting the environment information in the world, the robot can quickly and easily choose an action from the sensory information by referring to their preference values for every actions.

Section 2 will describe our definition and method. The simulated experiment is presented in Section 3. Section 4 will make a conclusion about our work.

2 Definition and Method

Let S be the state space, A be the set of actions, The *evaluation function* f is a mapping from the state-action pairs into real number, i.e. $f : S \times A \rightarrow \mathbf{R}$. Assume that each sensor affects the choices on actions independently, the function can be decomposed as $f = \Sigma_{i=1}^{n} f_i(s_i, a)$, where $s =< s_1, s_2, \cdots, s_n >$, and $f_i : S_i \times A \rightarrow \mathbf{R}$ is called the *preference value* for the ith sensor.

During the learning process, the preference values are adjusted according to the following procedure:

Sensor-differential learning algorithm

1 *Observing* ▷

Get sensory information s

2 *Choosing* ▷

Perform the action that maximizes $f(s, a)$

3 *Adjusting* ▷

$frequency(s_i) \leftarrow frequency(s_i) + 1, \forall s_i$

If the action succeeds,

$success(s_i, a) \leftarrow success(s_i, a) + 1, \forall s_i$
Total success count $\Phi \leftarrow \Phi + 1$

Otherwise

$failure(s_i, a) \leftarrow failure(s_i, a) + 1, \forall s_i$
Total failure count $\Psi \leftarrow \Psi + 1$

if $success(s_i, a) - failure(s_i, a) > 0$,

$$f_i(s_i, a) = \frac{\frac{success(s_i,a) - failure(s_i,a)}{frequency(s_i)}}{\Phi}$$

otherwise, $f_i(s_i, a) = -\frac{\frac{failure(s_i,a) - success(s_i,a)}{frequency(s_i)}}{\Psi}$

$$f(s, a) \leftarrow \alpha \cdot f(s, a) + (1 - \alpha) \cdot \Sigma_{i=1}^{n} f_i(s_i, a)$$

The evaluation function $f(s, a)$ is used for the robot to determine which action a is the most suitable action to perform while the current state is s. Initially, $f(s, a)$ is zero for each primitive action a and all the preference values of its sensor status are zero. The robot randomly selects an action to perform, and then rectifies the value of the evaluation function for this action. The rectification process depends on whether this action is succeeded in performing the behavior or not. Importantly, we make a *sensor differentiating process* for the sensor status in the rectification process, seen in the *adjusting* procedure of the algorithm. We keep recording the number of success and failure for each individual sensor status while the robot is performing an action, which are $success(\cdot, \cdot)$ and $failure(\cdot, \cdot)$. We divide them by the number of the sensor status that has happened to obtain the frequency of success and failure of that action. Then we divide the difference of those two frequency by the total success count to obtain the sensor status-action preference value if the frequency of success is bigger than the frequency of failure; otherwise, divide it by the total failure count.

The sensor status-action preference values are used to determine the sensor status classification. If we take the actions to be the concept on learning, the sensors will be the attributes for classification, and the sensor status are their values. For each sensor status and each action, there exists a sensor status-action preference value. The sensor status-action preference value demonstrates the degree that a sensor status belongs to a concept which is an action. A action is said to have the high probability to be chosen to perform when the robot state is with much more sensor status having high preference values of that action than the others, and the value of the *arbitration rate*, α, is much small.

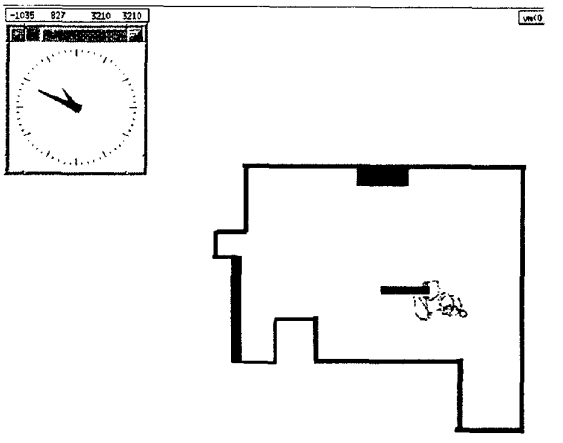

Figure 1: In the initial stage, the robot wandered in the map to learn the wall-following behavior

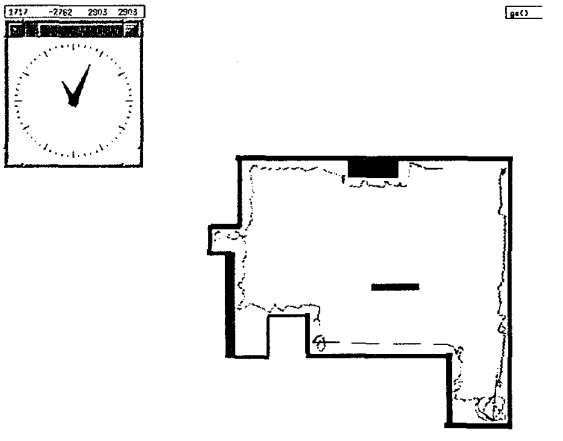

Figure 2: After about 15 minutes, the robot followed the left wall under a lower error rate.

At each moment in time, the robot gets information about the world from its sensors. According to those sensor status and the current sensor status-action preference values ($f_i(\cdot, \cdot)$, $\forall i$), we integrate all the values to determine which action is the most suitable and execute that action. The action which is most suitable for current robot state is the action with the maximum evaluation function value. If the robot succeeds in performing that action, all the sensor status-action preference values for the state will be increased by a reward; otherwise, all of them get a punishment. The preference values are then normalize for more efficient computation.

3 Experiment

In our experiments using a Nomad 200 mobile robot, 16 sonar sensors around the robot are taken. The sensor readings are divided into four states along each di-

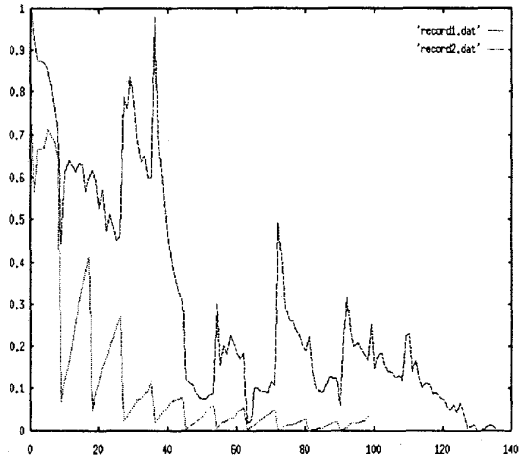

Figure 3: The error rates of two variants of our method.

rection. There are four primitive actions, moving forward, moving backward, turning right, and turning left, for the robot. The robot was trained to perform several behaviors, such as following the wall, moving along the corridor, and avoiding static obstacles in randomly generated simulation environment. After 15 training episodes on the SPARC IPX, the error rate can be reduced to 1% within ten minutes for the wall-following behavior. On average, each training experiment takes hundreds of steps when α is zero. The learned data can be used by the real robot with good performance. After learning, the role of each sensor status to each action is distinct.

For example, as seen in Figure.1 and Figure.2, without any prior knowledge of a given map and the rules of performing the wall-following behavior, the robot started from wandering in the map to following the wall by learning. Incrementally, the robot can correct the state-action function of doing the wall-following behavior. And inductively, the robot can extract the features of the robot sensors to deal with the wall-following behavior. It demonstrates that the robot can perform the wall-following behavior quite well within about fifteen minutes.

The error rate can be reduced to 1% within ten minutes by hundreds of steps, as shown in Figure.3 (The x-coordinate represents the number of learning phases. Each sample is taken at every ten steps of the robot operations.) On average, after about thirty minutes, it is almost nearly error-free. But the error rate is risen when the robot at a concave position, as seen in Figure.4.

The robot was always wandering at a small con-

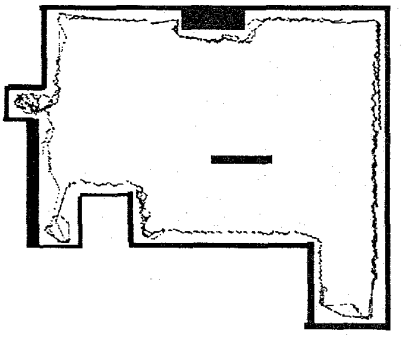

Figure 4: After about 30 minutes, the robot can perform the wall-following behavior quite well.

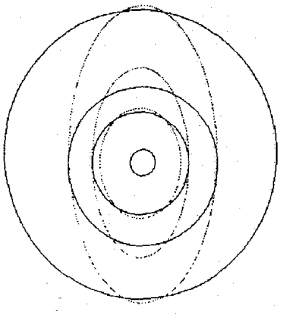

Figure 5: Partition range for robot's sensors.

cave area. That is because each sonic sensor of the robot is constantly partitioned into four status. All the sensor status can be drawn as four concentric circles around the robot as Figure.5. The robot situation is not enough to be represented by using the fixed partition. It needs a more precise partition. One is to increase the number of partition, but it will make the convergent rate of leaning increase. Since the learning speed is reverse proportional to the number of the robot states. The other is to make all robot sonar sensors with variable partition range. According to the sensors' characteristics for the wall-following behavior, we divide them into a fixed number but different partition boundary as the elliptical curves in Figure.5. As we known, different robot behaviors have their different characteristics. In order to learn different behaviors, we need to differentiate the robot sensors and make each robot sensor with different partition boundary for each behavior. It is not an easy and economic work. Dynamical adjustment may be a good idea for the robot behavior learning.

There are two variants of our method in this paper. Seen in Figure.3, they have very different learning performance. One is that we neglect the effect

of the state that is reappeared again in the current stage and the robot choose the same action to perform, we do not adjust the status-action preference function value of each robot sensor. The other is that we repeatedly increase the success function values of current robot sensory status, if the robot can succeed to execute the same action; otherwise, increase the failure function values of them. The former is much better than the latter. That is because the former can avoid the over-rewarding and over-punishing. Since we have given the reward or the punishment to the state-action function, so it is not necessary to distribute the effect of the success or the failure to each robot sensor. The time that needs to converge is also enlarged. The curves of the error rates of those two methods are fluctuating periodically. This is because the fixed partitions of the robot sensors are not enough to represent their situation for the wall-following behavior. That makes the robot stuck at local minimum of the state-action function at the concave region in our environment.

The table below shows part of the results from our experiments, which indicates that different values of α results in different convergence speeds and errors in the learned behavior. α is called the arbitration rate. It is used to adjust the reference degree between the values of the robot state-action evaluation function and the robot sensor status-action function.

Table. Performance of learning the wall-following behavior.

α	0.75	0.5	0.25	0
Average error rate after 300 steps	0.321	0.0234	0.247	0.0340
Elapsed time(sec) when error$\geq$ 0.01	916	623	399	190

The smaller the value of the α, the less the elapsed time when error $\geq 0,01$. But the vibration of the error rate is reverse proportional to the value of α. Because if α is approximate to zero means that the sensor status-action preference values are more important than the value of the previous state-action evaluation function. That is, the action selection policy almost full depends on the sensor status classification information. It is dangerous to determine the sensor status classification according to a few learning steps at the beginning. Those training examples are not enough to classify all the sixteen sonic sensor status. Some of sensor status will be classify to the wrong concept. More examples are needed to adjust it. In our experiments, it does not need much time to collect enough examples.

In our experiments, while we dynamically were adding or removing obstacles in our map, the robot was not burdened with the new situation, as shown in Figure.6. The robot can deal with the new environment according to its learned knowledge.

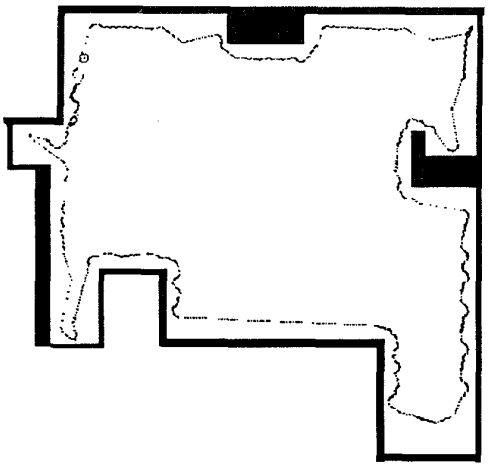

Figure 6: Dynamically add an obstacle to the environment.

4 Conclusion

This paper presents a new approach to reinforcement learning for the robot control that differentiates the roles of multiple sensors in action selection. The approach assumes that each sensor can independently contribute to the action selection policy.

By incrementally updating the preference value of each sensor to each action, the proposed method can

1) differentiate important sensors from irrelevant ones;

2) reduce the state space and thereby enabling training to converge quickly;

3) and achieve learned behaviors for the robot with low error rate.

Indirectly, the concept classification has been constructed by the sensor-differentiating process. Each action of a behavior is viewed as a concept. Associated with the reinforcement learning, the role of the sensor status for an action can be clearly differentiated by the sensor-differentiating process. Since the sonic sensors around our robot are independent, by our method, we can quickly and correctly identify their effects on performing that action without much space. The only data structure that needs to maintain is the sensor status-action preference value function. By integrating all the sensor status-action preference values for all actions, we can easily choose the most

suitable action.

In our experiments, to differentiate the roles of multiple sensors for a action is a helpful method for the robot control. It can quickly achieve the learned behaviors with low error rate. In our framework, all the sensors are taken to be independent, but how to do when some of them are relevant is an interesting problem. In our future work, we try to establish a incremental learning method for the robot control with the relevant sensors.

In the future, we try to add the classification of the relevant sensory information and the automatic adjustment of the sensors' partition for different behavior in our learning process. This paper has shown that the incremental learning by associating the conceptual classification with reinforcement learning for robot control can obtain a quite well performance.

References

[1] R.A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Rob. Autom.*, 2(1), 1986, pp. 14-23.

[2] R.A. Brooks, "Elephants don't play chess," *Robotics and Autonomous Systems*, 6, 1990, pp. 3-15.

[3] A.G. Barto, R.S. Sutton, and P.S.Brouwer, "Associative search network: a reinforcement learning associative memory," *Biological Cybernetics*, 40, 1981, pp. 201-211.

[4] D. Chapman and L. Kaelbling, "Learning from delayed reinforcement in a complex domain," in: *Proceedings IJCAI-91*, Sydney, NSW, 1991.

[5] L. Kaelbling, *Learning in embedded systems*, Ph.D. Thesis, Stanford University, Stanford. CA, 1990.

[6] L. Kaelbling, "Associative reinforcement learning: function in k-DNF," *Machine Learning*, 15, 1994, pp. 279-298.

[7] L. Kaelbling, "Associative reinforcement learning: a generate and test algorithm," *Machine Learning*, 15, 1994, pp. 299-319.

[8] L. Lin, *Self-improving reactive agents: case studies of reinforcement learning frameworks*, Tech. Rept. CMU-CS-90-109, Carnegie-Mellon University, Pittsbugh, PA, 1990.

[9] L. Lin, "Programming robots using reinforcement learning and teaching," in: *Proceedings AAAI-91*, Anaheim, CA, 1991.

[10] P. Maes and R.A. Brooks, "Learning to coordinate behaviors," in: *Proceedings AAAI-90*, Boston, MA, 1990, pp. 796-802.

[11] J.R. Quinlan, "Induction of decision trees," *Machine Learning*, 1991, pp. 338-342.

[12] R.S. Sutton, *Temporal credit assignment in reinforcement learning*, Ph.D. thesis, University of Massachusetts, Amherst, MA, 1984

[13] R.S. Sutton, "Learning to predict by the method of temporal differences," Machine Learning, 3(1), 1988, pp.9-44.

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

[15] C. Watkins, *Learning from delayed rewards*, Ph.D. Thesis, King's College, 1989.