

Programming Robots with Associative Memories

Claude F. TOUZET

Center for Engineering Science Advanced Research, Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge TN 37831-6355, USA, email: touzetc@mars.epm.ornl.gov

Abstract

Today, there are several drawbacks that impede the necessary and much needed use of robot learning techniques in real applications. First, the time needed to achieve the synthesis of any behavior is prohibitive. Second, the robot behavior during the learning phase is – by definition – bad, it may even be dangerous. Third, except within the lazy learning approach, a new behavior implies a new learning phase. We propose in this paper to use self-organizing maps to encode the non explicit model of the robot-world interaction sampled by the lazy memory, and then generate a robot behavior by means of situations to be achieved, i.e., points on the self-organizing maps. Any behavior can instantaneously be synthesized by the definition of a goal situation. Its performance will be minimal (not evidently bad) and will improve by the mere repetition of the behavior.

Introduction

Learning is a necessary component of robotics for reasons as serious as the time and money required to write ad-hoc behavior, or simply because an accurate-enough model of the environment is unavailable, as is the case of space exploration, sub-marine exploration, or nuclear powerplant assessment (after an accident).

The two most widely used robot learning paradigms are supervised learning and reinforcement learning. Supervised learning requires the operator to define a set of representative examples of situation-action pairs (i.e., the learning base). On the other hand, reinforcement learning generates the learning base through a combination of exploration and a reinforcement function. In the latter case, the operator is only asked to define a measure of the robot behavior performance. Despite the efforts to come up with a reinforcement function design process [7], a lot of time is spent in trial and error. Moreover, a reinforcement function has to be defined for each desired behavior, which means that – even if the reinforcement function is perfect – a new learning base must be build.

Lazy learning [1] reduces the time required to build the base. In an initial and unique sampling of the robot-environment relation, lazy learning builds a non-explicit model of the situation-action relation. Coupled to a reinforcement learning technique, such as Q-learning, lazy learning allows a great reduction of the time necessary for learning. The learning iterations are done in simulation using the non-explicit model, much faster than the actual time needed by a robot to performed the required number of actions. Lazy Q-learning [9] is becoming a paradigm of choice for robot learning, allowing almost instantaneous behavior synthesis - distributed lazy Q-learning techniques have already been proposed in the multi-agent context [3].

Lazy Q-learning application development is still time consuming. During development, many different reinforcement function expressions appear valid, and only experimentation is able to verify the quality of the synthesized behaviors. Research efforts in reinforcement function design only help the learning to converge, but there is no warranty that it will converge to the desired behavior. This is due to the highly indirect way the behavior is synthesized. A behavior is seen as a mapping between situation and action, and the learning is a function approximation method that uses generalization over a subset of high utility situation-action pairs, gathered during exploration [6]. The utility, initially a qualitative information, is transformed into quantitative values by the training rule. Being able to imagine *a priori* the behavior that will emerge from such complex process is very difficult, and progress in exploration, training rules, and generalization are not going to offer a definitive solution.

On the other hand, if the desired behavior is expressed not as a mapping between situations and actions, but as a situation to achieve (a goal to goal seek) then there is a direct relation (not necessarily a bijection) between the goal to be achieved and the representation of this goal in the operator's mind. Until today, goal-seeking methods in autonomous robotics have provoked little interest. They are mostly related to mapping applications, such as the go-to-the-nest application [8]. They associate a utility value with each situation-action pair encountered during the learning phase, which is later used as an indication of

which action to choose at a given location. If the position of the goal changes, then the learning must be started again. Applications to other domains, such as collision avoidance behavior, encounter the same limitation. Only one behavior is learned and the learned behavior cannot accommodate changes in the shape or size of the obstacle.

Using two self-organizing maps, we are able to provide the flexibility that is currently missing in goal seeking robot learning methods. The first self-organizing map builds a continuous representation of the situation space. This representation, for example a 2-D map, can be used to find a path – a set of intermediate situations – towards a goal situation, wherever it is. The second self-organizing map is used to generate the action to that will change the sensory inputs from the current perceived situation to the computed intermediate situation. The ability of the maps to represent the actual distribution of inputs allows for a greater discreteness in the areas of interest, i.e., the behavior performance will improve with repetition. Learning occurs by the mere realization of the behavior.

In the following section (section 2), we present the method used to build a behavior by locating intermediate situations to achieve on the self-organizing map. Experiments synthesizing an obstacle avoidance behavior for the Nomad 200 mobile robot are presented in section 3. Section 4 presents the related work and, finally, we conclude and offer a few ideas that extend and complete the learning method described in this paper.

2. Generating a behavior

We propose defining a behavior by means of a desirable goal for the robot to achieve. Therefore, the robot must be able to perceive such an achievement. The goal is then a robot sensory situation that is desirable. For example, it can be a perceived situation completely free of obstacles in the case of an obstacle avoidance behavior.

A behavior is a mapping between situations and actions. Dynamically, a behavior can be represented as a sequence of points in the situation-action space, each point belonging to the mapping. A sequence of points defines a trajectory, or a path. Generating the desired behavior is then producing the sequence of actions that will take the robot from its initial situation to the goal situation (assuming that the goal can not be achieved with only one action). The problem is that - due to the high dimensionality (usually much larger than 3) of the situation-action space – the number of situations for which an action has been tried is too small. Despite the use of lazy learning, the ratio of number situation-action pairs to search space size is extremely small. Therefore, for each current situation there are extremely few similar ones. This

number is also the number of actions among which we are able to choose the one that should take the robot closer to the goal. Most of the time, not enough situation-action pairs have been sampled to allow a reasonable selection - the number of points needed grows exponentially with the number of dimensions of the situation-action space. In conclusion, using lazy learning there is no guarantee that the selected course of action will lead the robot to the goal.

The use of self-organizing maps as associative memories allows a more aggressive course of actions. A self-organizing map (SOM) [5] is a clustering technique that adds a neighborhood property between the clusters. Clusters can be neighbors, or not. The number of neighbors per cluster gives the dimensionality of the map. For example, 4 neighbors correspond to a 2-D map. Unlike many clustering techniques, the number of clusters is fixed and predefined (at least in standard SOM versions). The positions of the clusters in the input space are defined by the input point distribution (Fig. 1a) and the neighbor's positions (Fig. 1b). The size of the clusters is directly related to the distribution. Each cluster corresponds to (approximately) the same number of input points. Therefore, regions where the input data are scarce will generate clusters with larger fields of attraction than regions of greater density.

The neighborhood conservation property allows the definition of a metric on the input space. If the input space is the situation space, then it becomes possible to (1) choose among clusters a neighbor situation closer to the

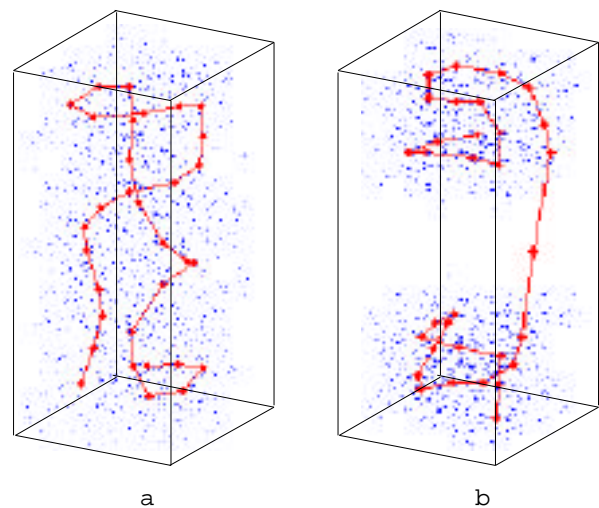


Fig. 1. The 32-cluster SOM maps the 3-D space into the 1-D SOM space (neighborhood of 2). Each cluster is represented by a circle and the lines drawn between clusters show the neighborhood. (a) The inputs are a set of uniformly distributed points (dots). (b) The inputs belong to 2 sets: the continuity property forces a few clusters to code space regions devoid of input points.

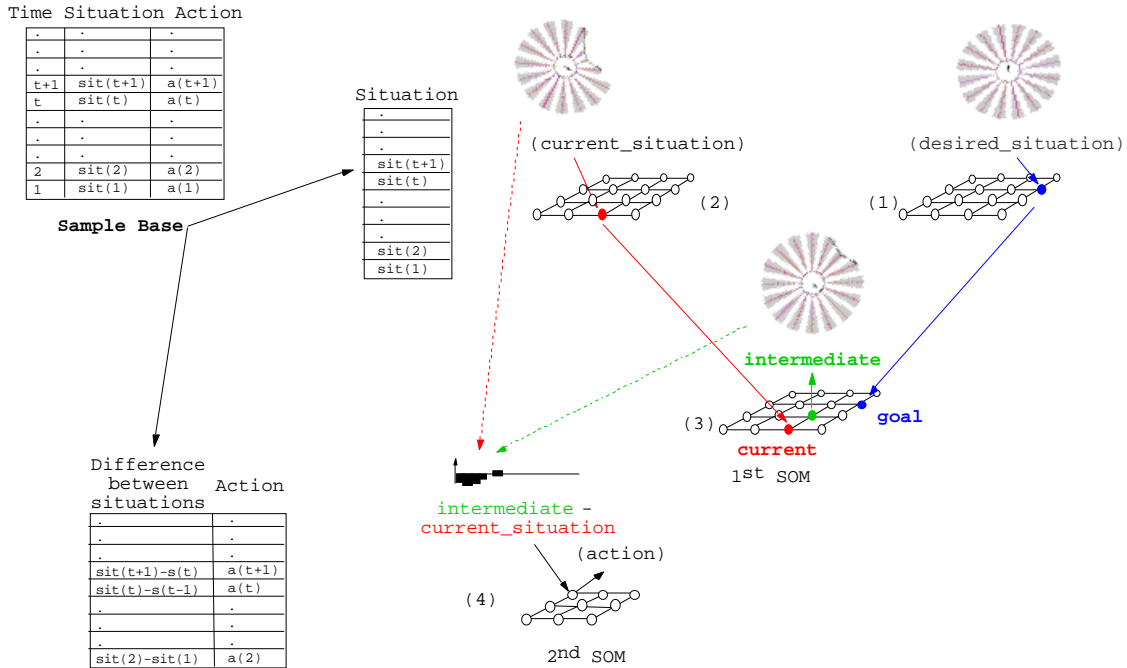


Fig. 2. The lazy memory (top left) is used to build the first SOM that maps the situation space, and the second SOM that maps the difference between sensory inputs versus the action. The samples of the learning base have been obtained using a random action selection policy for the robot. The behavior is generated through the following process: (1) Find the cluster corresponding to the desired goal situation. (2) Find the cluster corresponding to the current situation. (3) Find a neighbor cluster (of the current situation cluster) closer to the “goal” cluster which will represent the intermediate situation to achieve. (4) Probe the second SOM with the sensory variation between the intermediate cluster and the current situation to get the action that must be carried out.

goal, which will be the intermediate situation to achieve. Having found the intermediate achievable situation, we must now (2) obtain the action that will move the robot into that desired situation. This procedure (Fig. 2) must be repeated until the robot is in the desired goal situation (3). Fig. 3 shows the sequence of situations that occurred in an obstacle avoidance behavior generation.

3. Obstacle avoidance

Experiments have been conducted in synthesizing an obstacle avoidance behavior. The resulting behaviors do not bump into obstacles (Fig. 4) and are of a similar quality as the best results achieved with reinforcement learning [10], but need less learning iterations. The associative memories use the knowledge gathered by a random exploration of - only - 100 moves (compared to 200 minimum for Q-learning). And the same 100 iterations can be used to generate another behavior instantaneously, such as an obstacle avoidance that avoids by the right side. The goal is defined as a perceived free-of-obstacle sensory situation.

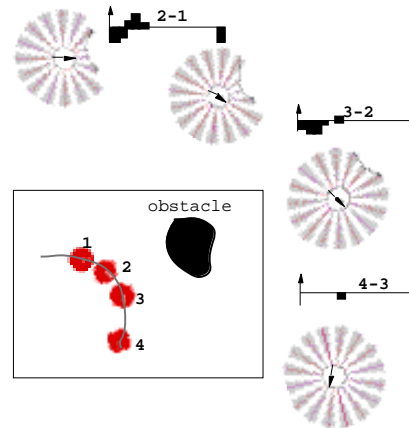


Fig. 3. Example of an obstacle avoidance sequence of situations. In the box, the successive positions of the robot and the obstacle are shown. The values of the 16 sensor sonar ring of the Nomad 200 are displayed. Also, the histograms show the sensory differences between two neighboring situations. These differences are the inputs for probing the second SOM to retrieve the actions.

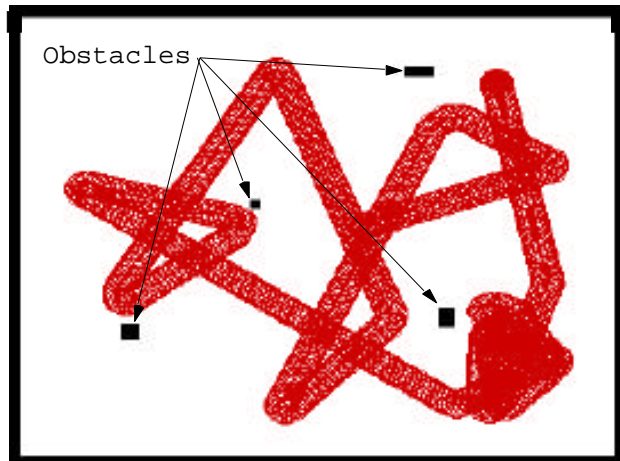


Fig. 4. Obstacle avoidance behavior generated by a Nomad 200 (mobile robot) using its infrared sensors (16). The position of the robot is indicated by a circle. The robot starts in the center of the area and moves towards the right. It avoids obstacles such as wall from a greater distance than the smaller obstacles – but it does not bump into any obstacle. The SOM use the knowledge gathered by a random exploration of – only – 100 moves. The goal is defined as a perceived free-of-obstacle sensory situation. Both SOMs comprise 16 clusters and a neighborhood of 4.

4. Related work

Coiton *et al.* [2] propose a neural network model for a sensorimotor system composed of a sensory layer (a SOM) and a motor layer. The objective is the generation of goal directed movements using a real robot arm. They show that their model is actually able to control the displacements of the robot arm. The input situations are three Cartesian coordinates and the outputs are three joint angles. The neural model learns the mapping between both sets of coordinates – only one behavior is possible. It is important to note that in our approach any behavior can be generated.

Conclusion

Using a set of two cooperating self-organizing maps, we have been able to demonstrate that any behavior – in fact any relation between situations and actions – can be generated. This solves the problem stated in the introduction: immediate behavior generation with goal seeking methods. Many different behaviors can be obtained using the same SOM, by simply defining other goal situations, such as wall following or following

another robot. The defined goals are independent of the robot geometry or actuators. The amount of supervision required by the human operator is minimal compared to other approaches like supervised learning or reinforcement learning.

More than one sensory modality can participate in the expression of a behavior. In our framework, every sensory modality will be taken care of by a couple of SOM. These maps, carrying diverse representations, must be put together from time to time (at least to generate intelligent behaviors). As stated by Gallistel [4], space and time are two predominant aspects of reality and must therefore be part of any stored record (i.e., situation-action pair). These two coordinates link the separate records within the same SOM and between SOMs. This linking between records is what enables complex behaviors to occur. Note that the linking of records is accomplished *a posteriori*, at the time of the retrieval and that there is no implication of the retrieval process in the memorization process. A desired behavior is generated by positioning a goal in a map, whose spatial and/or temporal coordinates will then be used to retrieve other records in the same or different maps, generating automatically a sequence of sub-goals to achieve, i.e., the action plan.

References

- [1] Aha D. (ed.), *Lazy Learning*, Kluwer Academic Pub., 1997.
- [2] Coiton Y., J. C. Gilhodes, J. L. Velay and J. P. Roll, "A neural network model for the intersensory coordination involved in goal-directed movements," *Biological Cybernetics*, 66, 1991, 167-176.
- [3] Darrell, T., "Reinforcement Learning of Active Recognition Behaviors," Interval Research Technical Report 1997-045. (<http://www.interval.com/papers/1997-045> - Portions of this paper previously appeared in *Advances in Neural Information Processing Systems 8*, (NIPS '95), pp. 858-864, MIT Press, and *Intelligent Robotic Systems*, M. Vidyasagar ed., pp. 73-80, Tata Press, 1998.
- [4] Gallistel R., *The Organization of Learning*, MIT Press, 1990.
- [5] Kohonen T., *Self-Organization and Associative Memory*, Second Edition, Springer Series in Information Sciences, Vol. 8, Springer Verlag, Berlin, 1987.
- [7] Santos J. M. and C. Touzet, "Exploration Tuned Reinforcement Function," *Neurocomputing*, Spring 1999.
- [8] S. Sehad and C. Touzet, "Neural Reinforcement Path Planning for the Miniature Robot Khepera," *WCNN'95*, Washington D.C., USA, 17-21 July 1995.
- [9] Sheppard J. W. and S. L. Salzberg, "A Teaching Strategy for Memory-Based Control," *Lazy Learning*, D. Aha (ed.), Kluwer Academic Publishers, 343-370, 1997.
- [10] Touzet C., "Neural Reinforcement Learning for Behaviour Synthesis," Special issue on Learning Robot: the New Wave, N. Sharkey (guest ed.), *Robotics and Autonomous Systems* 22, No 3-4, 251-281, 1997.