Latent Learning in Agents

Rati Sharma RATIS@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, NJ 08854, USA

Abstract

Various cognitive models have been proposed to determine optimal paths in spatial navigation tasks, some of which demonstrate latent learning in Agents. We view and present the model as a Reinforcement Learning problem by using Q-Learning and Model-Based Learning in a deterministic environment. This paper uses a Q-Learning algorithm and compares its performance with the Dyna (Model based) algorithm on Blocking and Shortcut problems. We conclude that using Model-Based Reinforcement learning provides an interesting and efficient solution to blocking and shortcut problems and that the agent demonstrates Latent Learning.

1. Introduction

An important task for autonomous agents including humans, animals and robots is that of spatial navigation. How does a robot decide what to do? How does it determine the optimal path from point A to B? The traditional answer has been that the robot should be able to plan but one of the serious limitations of planning is that the robot should have an accurate world model. We describe an approach where the robot learns a good set of reactions by trial and error and at the same time plans to generate an internal model of the world. This internal model can be considered similar to a Cognitive Map that is compiled during exploration of the environment.

Tolman (1948) proposed a theory based on his experiments performed on rats in a T maze (introduced by Blodget) that the animals form cognitive maps i.e. in the course of learning, a field map of the environment gets established in the rat's brain. Cognitive maps store the expectations that when situated in place *A*, if the animal executes response *R* (a movement in a given direction), it will end in place *B*. This Cognitive Map that represents routes, paths and environmental relationships determines the response R.

One form of spatial navigation that incorporates the idea of a Cognitive Map has been presented by Voicu and Schmajuk (2000). The model is presented as a neural network that assumes a representation of the environment as a set of connected locations. As part of developing a Cognitive Map, a representation of the structure of the actual environment was stored in the map, which was then used to guide navigation between specific places. We view robot spatial navigation as a reinforcement learning problem and use Q-Learning and Model-Based Reinforcement Learning methods.

Reinforcement learning is learning what to do, how to map situations to actions leading maximization of the rewards. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trial and error interaction with the environment.

2. Latent Learning

Latent learning is described as learning that is not evident to the observer at the time it occurs and that is apparent once a reinforcement is introduced. Here is an explanation of latent learning as it occurs in rats. A T-maze proposed by Blodget (Figure 1) was used by Tolman to demonstrate latent learning in Rats. The rats were allowed to explore the maze without any reward (food) for several trials. However once a reward was introduced, the rats with great accuracy could reach the goal (food) in the maze. It was proposed that as long as the animals were not getting any food at the end of the maze they continued to take their time in going through it, they continued to enter many blinds. Once, however, they knew they were to get food, they demonstrated that during these preceding non-rewarded trials they had learned where many of the blinds were. They had been building up a 'map,' and could utilize the latter as soon as they were motivated to do so, by the introduction of a reward.

For the purpose of our simulation we use a slight variation of the above T-maze (Figure 2).

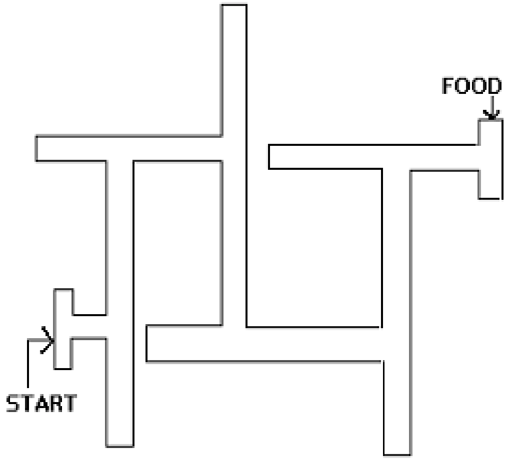

Figure 1. T-Maze used by Tolman (From H.C. Blodget, The effect of the introduction of reward upon the maze performance of rats. University of California Publications in Psychology, 1929, 4, p. 117)

2. A Navigation Task

The T-maze in Figure 2 depicts a typical maze used for the navigation experiments. The maze is a four by seven grid of possible locations or states, one of which is marked as the starting state (in yellow) and one of which is marked as the Goal state (in green). The hurdles or barriers in the grid are also marked (in red). From each state there are four possible actions UP, DOWN, RIGHT and LEFT, which change the state except when the resulting state would end up in a barrier or take the agent out of the bounds of the maze. The reward for all states is 0 except for the goal state where it is 1. The environment is considered to be deterministic.

In the following section we describe the two learning algorithms along with their simulations.

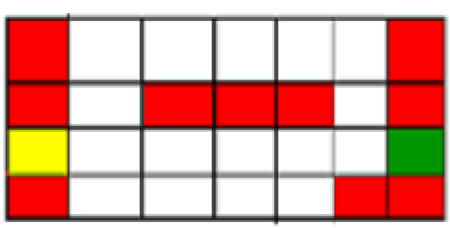

Figure 2 The grid world used for computer simulations

3. Learning Algorithms

The two reinforcement learning algorithms used for this purpose are described as below:

3.1 Q-Learning

Q-Learning was proposed by Watkins (1989). The main idea in the algorithm is to approximate the function assigning each state-action pair the highest possible payoff, thus for each pair of state *s* and action *a*, Q-learning maintains an estimate *Q(s, a)* of the value of taking *a* in *s*. The agent does not have a model of the world in advance and it tries to create an optimal policy while learning a world model.

Let *Q(s, a)* be the evaluation function so that its value is the maximum discounted reward that can be achieved starting from state *s* and applying action *a* as the first action i.e. the value of *Q* is the reward received immediately upon executing action *a* from state *s*, plus the discounted value of following the optimal policy thereafter.

$Q(s, a) = r(s, a) + \gamma * max\ Q\ (\delta(s, a), a')$

r(s, a) is the reward function,

δ(s,a) is the new state.

Q learning algorithm:

Q^(s, a) refers to the learner's current hypothesis about the actual but unknown value *Q(s, a)*

For each *s*, *a* initialize the table entry *Q^(s, a)* to zero.

Observe the current state s

Do Forever:

1) Select an action a and execute it
2) Receive immediate reward *r*
3) Observe the new state *s'*
4) Update the table entry for *Q^(s, a)* as follows:

$Q\hat{}(s, a) \leftarrow r + \gamma\ max\ Q\hat{}(s', a')$

3.2 Model-based Learning

The agent tries to learn the world-model in advance and then utilizes the plan to generate navigation plans. In this paper we work with the Dyna class of reinforcement learning architectures proposed by Sutton (1990). The Dyna architecture is based on dynamic programmings policy iteration. It simultaneously learns by trial and error, learns a world model and plans optimal routes using an evolving world model.

3.2.1 Overview of Dyna Architecture

Dyna architectures use machine learning algorithms to approximate the conventional optimal control technique i.e. Dynamic Programming. Dynamic programming is a computational method for determining optimal behavior given a complete model of the task to be solved. Thus, Dyna performs a form of planning in addition to learning. While learning updates the value function estimates according to experience as it actually occurs, planning differs only in that it updates these same value function estimates for simulated transitions chosen from the world model.

The Dyna architecture consists of 4 components, as shown in Figure 3. The *policy* is the function formed by the current set of reactions; it receives as input a description of the current state of the world and produces as output an action to be sent to the world. The *world* represents the environment of the robot. It receives actions from the policy and produces as output the next state of the agent and a reward. The *world-model* in the architecture intends to mimic the one step behavior of the real world. The *reinforcement learner* represents a learning algorithm like Q-Learning.

The Dyna algorithm:

a) Decide if this will be a real experience or a hypothetical one.

b) Pick a state x (in a real experience choose a current state).

c) Choose an action:

$Q(x, a) = E(r+\gamma e(y) \mid x, a\}$ i.e. the Q value for a state x and an action a is the expected value of the immediate reward r plus the discounted value of the next state y.

d) Do action a, obtain the next state y and reward r from world or the world-model.

e) If this is a real experience, update world model from x, a, y and r.

f) Update the evaluation function and update the policy, i.e. strengthen or weaken the tendency to perform action a in state x according to the error in the evaluation function.

g) Go to Step 1.

This algorithm takes advantage of the fact that exploration at regular intervals provides a better chance of determining an optimal policy as opposed to not exploring once learning has occurred. This will be evident in the computer simulations.

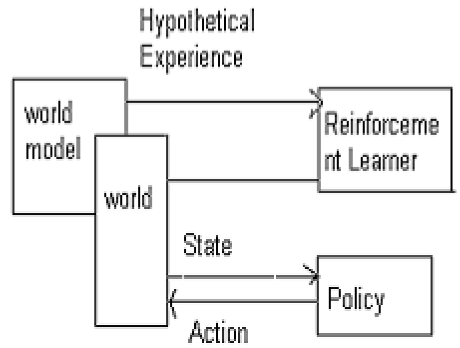

Figure 3 An overview of the Dyna architecture. The reinforcement learner represents an algorithm like Q-Learning.

The initial Q values, the policy table entries and the world model were all empty and the initial policy was a random walk. For our experiments real and hypothetical experiences were used alternately. For each experience with the real world, k hypothetical experiences were generated with the model.

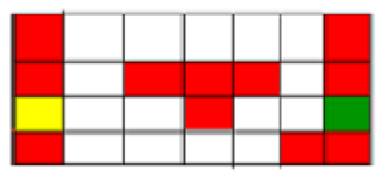

Figure 4 The grid world for the blocking problem.

4. Computer Simulations:

<u>*Problems considered*</u>: We perform computer simulations for two problems namely the blocking problem, where by after learning an optimal path from the start to the goal state a block in introduced in the path, forcing the agent to consider alternate paths of navigation. The shortcut problem represents the opposite approach, in which once again after the agent has learnt the optimal path from the start state to the goal, a shortcut is introduced, and it is observed whether the agent performs any exploration and is able to find the shortcut, therefore, improving over the previous optimal path.

We evaluate Q-Learning and Model-based learning in a deterministic (grid) environment (Figure 4) for the *blocking problem*. The default values for the various parameters are: Discount factor γ = .9, learning rate = 0.5

The resultant graph depicting Q Learning and Model - Based Learning is shown in Figure 5.

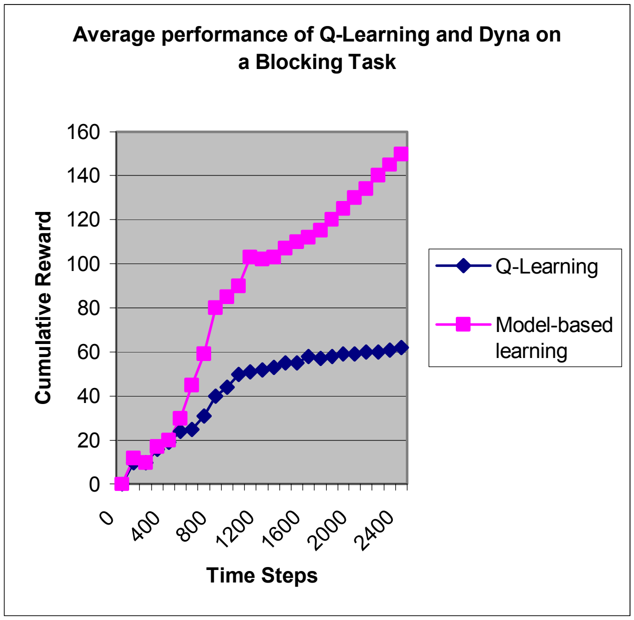

Figure 5 A graph depicting average performance over 15 runs for the Dyna and Q-Learning algorithms

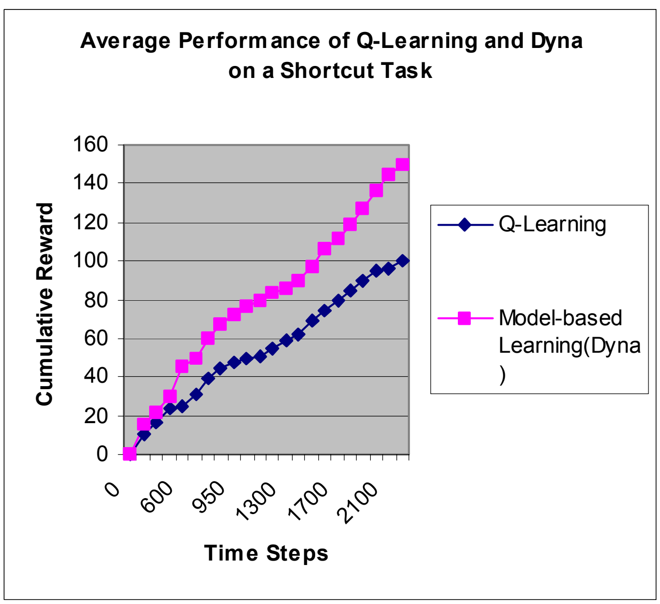

Figure 6 A graph depicting the average performance of the Algorithms on a Shortcut problem.

5. Analysis and Conclusion

Q-Learning when combined with sufficient exploration can be guaranteed to eventually converge to an optimal policy without ever having to learn an use an internal model of the environment. However, the cost of not using an internal model is that convergence to the optimal policy can be very slow and in a large state space many backups may be required before the necessary information is propagated to where it is important. From our experiments it is evident that the optimal state value function is highest near the goal and declines exponentially as the number of steps required to reach the goal increases. The Dyna architecture provides a solution to this problem as it generates and maintains an internal world-model; also the model facilitates the proper exploration of the environment and as is evident from the results performs much better in dynamically changing environments.

We conclude based on these experiments that reinforcement learning architectures such as Dyna exhibit latent learning. As the world model changes they are able to adapt and recover more quickly than model free approaches such as Q-Learning. This behavior is comparable to that of rats when placed in a maze. Tolman observed in several of his experiments that once latent learning had occurred in rats, and later on if barriers or short cuts were introduced in the environment rats were able to adapt quickly and subsequently determine the optimal paths to the goal.

6. Future Work

We worked with a deterministic grid environment in this paper; it would be interesting to see the performance of these approaches in a non-deterministic world. Also, the use of a probabilistic approach in determining hypothetical experiences in Dyna would probably yield more efficient results.

Acknowledgements

This project would not have been possible without the guidance of Professor Littman. I sincerely thank him for his patience and encouragement.

References

Kaelbling, L., Littman, M., Moore, A. (1996) Reinforcement Learning: *A survey. Journal of Artificial Intelligence Research, 4, 237-285.*

Sutton, R. (1990) Integrated Architectures for Learning, Planning and Reacting based on Approximate Dynamic Programming. *Appeared in Proceedings of the Seventh International Conference on Machine Learning, pp. 216-224, Morgan Kaufmann.*

Tolman, E.C. (1948). Cognitive maps in Rats and Men. *The Psychological Review. 55(4), 189-208*

Voicu, H., Schmajuk, N. (2002). Latent Learning, shortcuts and detours: a computational model. *Behavioral Processes 59 (2002) 67-86.*