Dual Memory Model for Using Pre-Existing Knowledge in Reinforcement Learning Tasks

Kary Främling

Helsinki University of Technology, PL 5500, FI-02015 TKK, Finland
Kary.Framling@hut.fi

Abstract. Reinforcement learning agents explore their environment in order to collect reward that allows them to learn what actions are good or bad in what situations. The exploration is performed using a policy that has to keep a balance between getting more information about the environment and exploiting what is already known about it. This paper presents a method for guiding exploration by pre-existing knowledge expressed as heuristic rules. A dual memory model is used where the value function is stored in long-term memory while the heuristic rules for guiding exploration act on the weights in a short-term memory. Experimental results from a grid task illustrate that exploration is significantly improved when appropriate heuristic rules are available.

1 Introduction

Supervised and unsupervised learning construct models that represent training samples given by a "teacher" as well as possible. Reinforcement learning (RL) methods differ from these learning methods because the RL agent has to explore its environment by itself and collect training samples. Initially the agent has to take actions without knowing how good or bad they are, which may be known only much later when a reward signal is provided. The agent takes actions following a policy, usually referred to by the symbol π that should make it explore the environment efficiently enough to learn at least a near-optimal policy.

This paper presents a new method for using pre-existing knowledge for state-space exploration. It uses a dual memory model where a so-called long-term memory is used for action-value learning and a short-term memory is used for modifying action selection by heuristic rules. Experimental results with simple heuristic rules show that it is possible to converge to a good policy with less exploration than with some well-known methods.

After this introduction, the most relevant RL methods to the scope of this paper are described in Section 2. Section 3 presents methods to guide exploration, while Section 4 shows comparative test results for grid world tasks, followed by conclusions.

2 Reinforcement learning priciples

One of the main domains treated by RL is Markov Decision Processes (MDPs). A (finite) MDP is a tuple $M=(S,A,T,R)$, where: S is a finite set of states; $A = \{a_1, \ldots, a_k\}$ is a set of $k \geq 2$ actions; $T = [P_{sa}(\cdot) \mid s \in S, a \in A\}$ are the next-state transition probabilities, with $P_{sa}(s')$ giving the probability of transitioning to state s' upon taking action a in state s; and R specifies the reward values given in different states $s \in S$. RL methods try to learn a value function that allows them to predict future reward from any state when following a given action selection policy π. Value functions are either *state-values* (i.e. value of a state) or *action-values* (i.e. value of taking an action in a given state). Action-values are denoted $Q(s,a)$, where $a \in A$.

The currently most popular RL methods are so-called *temporal difference* (TD) methods. *Q-learning* is a TD control algorithm that updates action values according to

$$\Delta Q(s_t,a_t) = \beta\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right] e_{t+1}(s) \qquad \textbf{(1)}$$

where $Q(s_t, a_t)$ is the value of action a in state s at time t, β is a learning rate, r_{t+1} is the immediate reward and γ is a discount factor that determines to what degree future rewards affect the value. The max-operator signifies the greatest action value in state s_{t+1}. $e_{t+1}(s)$ is an eligibility trace that allows rewards to be propagated back to preceding states.

Exploration of the state space is performed by *undirected* or *directed exploration* methods [3]. Undirected exploration methods do not use any task-specific information, e.g. *ε-greedy exploration* (take greedy action with probability (1-ε) and an arbitrary action with probability ε) and *Boltzmann* action selection (selection probabilities proportional to action values). Directed exploration methods use task-specific knowledge for guiding exploration in such a way that the state space would be explored more efficiently. *Counter-based* methods direct exploration towards states that were visited least frequently in the past, *recency-based exploration* prefers states that were visited least recently while other directed exploration methods use confidence estimates for the current value function [2]. The well-known technique *optimistic initial values* uses initial value function estimates that are bigger than the expected ones, therefore privileging unexplored actions (and states). Initializing action values to zero and giving negative reward for every action are often used to implement this technique [1].

3. SLAP reinforcement and the BIMM network

This section describes the *bi-memory model* (BIMM) that uses a *short-term memory* (STM) for controlling exploration and a *long-term memory* (LTM) for action-value learning. Both memories use linear function approximation by the one-layer Adaline artificial neural net (ANN) [4] but any function approximator could be used for both STM and LTM. Weights are stored in a two-dimensional matrix of size $P \times N$, where P

is the number of actions and N is the number of state variables. If only one state variable is allowed to be one while all others are zero this representation is identical to the lookup-table representation usually used in discrete RL tasks. A major advantage of using a function approximator approach over lookup-tables is that they can handle continuous-valued state variables. BIMM outputs are calculated according to

$$a_j(s) = K_0 \sum_{i=1}^{N} ltw_{i,j} s_i + K_1 \sum_{i=1}^{N} stw_{i,j} s_i \tag{2}$$

where $a_j(s)$ is the estimated action-value and K_0 and K_1 are positive constants that control the balance between exploration and exploitation as in [2]. $ltw_{i,j}$ is the LTM weight and $stw_{i,j}$ is the STM weight for action neuron j and input i. Q-learning or some other action-value learning method can be used to update LTM weights by replacing Q with ltw in equation (1). STM is an exploration bonus that decreases or increases the probability of an action being selected in a given state and whose influence on action selection is determined by the value of K_1. In the tests performed in this paper, action probabilities are only decreased by the *SLAP* (Set Lower Action Priority) method according to heuristic rules (see test section). SLAP updates STM weights using the Widrow-Hoff rule with the target value

$$a_j'(s) = a_{min}(s) - margin \tag{3}$$

where $a_{min}(s)$ is the smallest $a_j(s)$ value in state s[1]. Only STM weights are modified by the Widrow-Hoff rule, which becomes

$$stw_{i,j}^{new} = stw_{i,j} + \alpha(a_j' - a_j)s_i \ . \tag{4}$$

where α is the learning rate The new activation value is

$$\begin{aligned} a_j^{new}(s) &= K_0 \sum_{i=1}^{N} ltw_{i,j} s_i + K_1 \sum_{i=1}^{N} stw_{i,j}^{new} s_i = \\ K_0 \sum_{i=1}^{N} ltw_{i,j} s_i &+ K_1 \sum_{i=1}^{N} s_i \left(stw_{i,j} + \alpha(a_j' - a_j)s_i \right) = a_j(s) + \alpha \sum_{i=1}^{N} s_i^2 K_1 (a_j' - a_j) \end{aligned} \tag{5}$$

where we can see that setting α to $1/(\Sigma s_i^2 K_1)$ guarantees that a_j^{new} will become a_j' in state s after SLAPing action j. This is a generalization of the well-known Normalized Least Mean Squares (NLMS) method, so from now on α will systematically refer to α_{norm} in BIMM. In stochastic tasks, α should be inferior to $1/(\Sigma s_i^2 K_1)$ because even the optimal action may not always be successful, so immediately making it the least attractive is not a good idea. A general algorithm for using SLAP in a learning task is given in Fig. 1.

[1] The *margin* should have a "small" value that ensures that an action that is repeatedly SLAPed will eventually have the lowest action value. 0.1 has been used in all tests.

```
Initialize parameters
REPEAT (for each episode)
  s ← initial state of episode
  REPEAT (for each step in episode)
    a ← action given by π for s
    Take action a, observe next state s′
    SLAP "undesired" actions
    Update action-value function in LTM
    s ← s′
```

Fig. 1. General algorithm for using SLAP in typical RL task

3.3 Increasing exploration

As shown for the experimental tasks in section 4, heuristic rules can make exploration faster but they may not guarantee a sufficient exploration of the state space. Using an undirected exploration method in addition to the heuristic rules can compensate for this. In the tests reported in this paper, STM weights have been initialized to random values in the interval [0,1) while LTM weights are initialized to zero. Therefore actions will be selected in a random order as long as LTM weights remain zero and no action has been SLAPed. When LTM weights become none-zero, the degree of randomness depends on the value of K_I. Setting K_I to a small value will give no or little randomness (10^{-6} has been used in the tests of section 4) while a greater value will give more randomness. In deterministic tasks it is easy to show that setting $K_I = 1.0$ and always SLAPing the greedy action from equation (2) will perform a depth-first search of the state space.

4. Experimental results

This section compares different methods on a typical maze task (Fig. 2) with four possible actions. Both deterministic and stochastic state transition rates 0.2 and 0.5 (the probability of another direction being taken than the intended one) are tested.

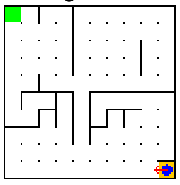

Fig. 2. Maze. Agent is in start state (lower right corner), terminal state in upper left corner

The compared methods are: **1)** ***Q***: ε-greedy exploration, zero initial Q-values, $r = 1.0$ at terminal state and zero for all other state transitions; **2)** ***OIV***: optimistic initial values, zero initial Q-values, $r = 0.0$ at terminal state and $r = -1.0$ for all other state transitions; **3)** ***BIMM***: zero initial Q-values (LTM weights), r = 1.0 at goal and zero for all other state transitions and **4)** ***CTRB***: counter-based exploration, zero initial Q-values, r = 1.0 at goal and zero for all other state transitions. Q-learning without eligibility traces is used by all methods for action-value learning. Learning parameters are indicated in Table 1. In deterministic tasks $\beta = 1$ is always the best learning rate for Q-learning [1].

The counter-based exploration bonus is implemented as

$$\delta_1(s,a) = \begin{cases} 1.0 - \dfrac{cnt(s,a) - cnt(s)_{\min}}{cnt(s)_{\max} - cnt(s)_{\min}} & \text{if } cnt(s)_{\max} - cnt(s)_{\min} > 0 \\ 0.0 & \text{if } cnt(s)_{\max} - cnt(s)_{\min} = 0 \end{cases} \qquad \textbf{(6)}$$

where $cnt(s,a)$ is the counter for action a in state s and $cnt(s)_{min}$ and $cnt(s)_{max}$ are the smallest and greatest counter values for all actions in state s. Counter values and BIMM STM weights are reset before beginning a new episode. SLAP was used according to the following rules when entering a new state and before taking the next action: 1) SLAP the “inverse” action and 2) if the new state is already visited during the episode, SLAP action with the biggest value $a_j(s)$ in equation (2) for the new state. For BIMM, using ε-greedy exploration in the deterministic task converged to a better policy than when using a “high” K_I-value (e.g. 0.1). In the stochastic tasks such supplementary exploration was not needed. Parameter values are indicated in Table 1.

Table 1. Parameter values used in grid world tests. $K_I = 10^{-6}$ for BIMM in all tasks. All parameters not indicated here were set to zero

Agent	***Q, γ=0.95***		***OIV***		***BIMM, γ=0.95***			***CTRB, γ=0.95***	
Task	β	ε	β	γ	α	β	ε	β	K_I
Deterministic	1	0.1	1	1	0.1	1	0.1	1	0.1
Stochastic, 0.2	0.1	0.1	0.5	0.95	0.2	0.5	0.0	0.5	0.01
Stochastic, 0.5	0.1	0.1	0.5	0.95	0.1	0.5	0.0	0.5	0.01

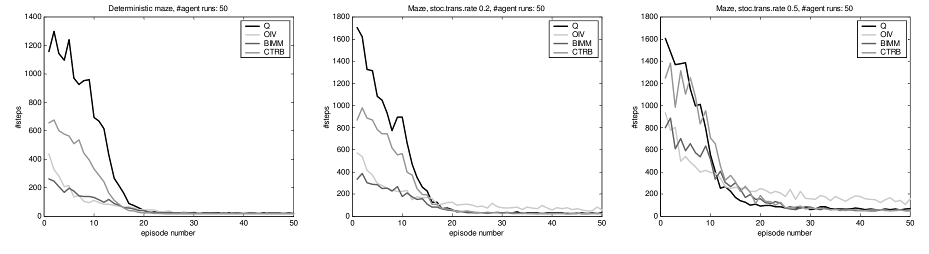

Fig. 3. Maze results as average number of steps per episode. Stochastic transition rate and the number of agent runs used for calculating the average number of steps are indicated at the top of each graph

All agents performed 250 episodes. After 200 episodes actions were selected greedily using the learned action-values by setting ε and K_I to zero for all agents. From Fig. 3 it is clear that BIMM converges towards a “good” policy after less exploration than Q- and CTRB-agents. Only the OIV agent can compete in the beginning of exploration but it converges very slowly. With stochastic transitions OIV fails to converge due to the increased probability of cycles during exploration, which causes the action-values to be continually modified due to the negative step reward.

As indicated by the first column in Table 2, on the first episode BIMM agents reach the terminal state faster than all other agents. BIMM also represents the best compromise between how good the "converged" policy is and how much exploration is needed. BIMM has the best converged policy or is very close to it in all tasks while using the smallest total number of steps in both stochastic tasks.

Table 2. Maze results. Numbers are averages from 50 agents runs, indicated as Q/OIV/BIMM/CTRB. Steps with "converged" policy are averages of episodes 241-250. STR: stochastic transition rate

Task	**Steps first episode**	**Steps conv. policy**	**Total nbr. of steps**
Deterministic	1160/438/**263**/654	18.0/18.0/18.0/18.2	17700/**7070**/7380/10600
STR 0.2	1700/571/**335**/871	**24.0**/26.7/24.4/25.1	20300/13600/**9730**/15600
STR 0.5	1610/937/**798**/1250	45.2/50.7/**44.3**/46.1	26100/26400/**21300**/25800

5. Conclusions

The results show that with appropriate heuristic rules for guiding exploration, BIMM and SLAP can improve exploration both in deterministic and stochastic environments. BIMM agents only use their own general domain knowledge, which makes them interesting compared with methods like reward shaping that usually require a priori knowledge about the task itself and easily lead to sub-optimal solutions.
Even though only a grid world task is used in this paper, BIMM and SLAP are also applicable to tasks involving continuous-valued state variables. Such tasks are a subject of current and future research.

References

1. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, Vol. 4 (1996) 237-285
2. Ratitch, B., Precup, D.: Using MDP Characteristics to Guide Exploration in Reinforcement Learning. In: Lecture Notes in Computer Science, Vol. 2837, Springer-Verlag, Heidelberg (2003) 313-324
3. Thrun, S.B.: The role of exploration in learning control. In: DA White & DA Sofge (eds.): Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches. New York, Van Nostrand Reinhold (1992)
4. Widrow, B., Hoff, M.E.: Adaptive switching circuits. In: 1960 WESCON Convention record Part IV, Institute of Radio Engineers, New York (1960) 96-104