Liet. matem. rink., **44**, *spec. nr.*, 2004, 797–801

Making virtual learning environment more intelligent: application of Markov decision process

Dalia BAZIUKAITĖ (KU)
e-mail:dalia@ik.ku.lt

Abstract. Suppose there exist a Virtual Learning Environment in which agent plays a role of the teacher. With time it moves to different states and makes decisions on which action to choose for moving from current state to the next state. Some actions taken are better than some others. The transition process through the set of states ends in some final (goal) state, being in which it gives for the agent the largest benefit. The best way of action is to reach the goal state with maximum return available. The system is formalized as Markov Decision Process and the Q-Learning algorithm is applied to find of such kind criterion that optimises the behavior of the agent.

Keywords: Markov decision process, reinforcement learning, virtual learning environment, Q-learning.

1. Introduction

Virtual learning environment (VLE) is defined to be the web-based application suitable for information exchange between teacher and learner [4], as well as between learners themselves. There exist another subset of VLEs that differs from the previous in ability of sequencing the curriculum. Those VLEs are known as adaptive ones [1, 2, 5]. The adaptivity is implemented in the systems that are based on the aproach used in the ELM-ART II [6] adaptive tutoring system that supports learning programming in LISP. One more subset of VLEs is intelligent environmets. Intelligent environments use the methods of Artificial Intelligence to solve the problem of student modeling and curriculum sequencing [12]. In the intelligent VLE the role of teacher is prescribed to the agent, who makes decisions how to teach the learner depending on the experience presented to the agent.

The reinforcement learning is the process during which the agent improves its behavior in the environment [7, 8, 9, 11] using the experiences it has got interacting with an environment. The experiences have the form of tuple $\langle x, a, y, r\rangle$, with state x, action a, resulting state y and scalar immediate reward r. The Markov Decision Process (MDP) model is a way of formalizing the reinforcement learning problem. A finite MDP is defined by the tuple $\langle S, A, P, R\rangle$, with a finite set of states S, a finite set of actions A, a transition function P, and a reward function R. The optimal behavior for an agent in an MDP depends on an optimallity criterion. The optimality criterion could be expressed as infinit-horizon expected discounted total-reward [7, 8] or infinit-horizon expected average reward [9, 10]. For those both cases the optimal behavior can be

found by identifying the optimal value function defined recursively by

$$V^*(x) = \max_a \left(R(x,a) + \gamma \sum_y P(x,a,y) V^*(y) \right) \tag{1.1}$$

in case of infinit-horizon expected discounted total-reward, where $0 \leqslant \gamma < 1$ is a discount parameter that controls the degree to which future rewards are significant compared to immediate rewards, or by

$$V^*(x) = \max_a \left(R(x,a) - \rho^* + \sum_y P(x,a,y) V^*(y) \right), \tag{1.2}$$

in case of infinit-horizon expected average reward, where ρ^* is an average reward.

2. Markov decision processes

Equations (1.1) and (1.2) are called Bellman's equations for discounted reward and average reward case correspondingly. In the rest of the paper we will concentrate on the case of the finit-horizon expected discounted reward. Bellman's equation formalizes the overall goal of the Markov Decision Process. The goal is to find a function, called a policy, which specifies which action to take in each state, so as to maximize some function of the sequence of rewards [10, 11]. Markov Decision Process is defined by a set of possible states of the agent $X = \{x_1, x_2, ..., x_n\}$, a set of actions $A = \{a_1, a_2, ..., a_m\}$, a set of rewards $R = \{r_1, r_2, ..., r_n\}$ and a transition probability function $P_{ij}^k = Pr\{x_{t+1} = y | x_t, a_t\}$, where $i, j = 1..n$, $i \neq j$, and $k = 1..m$. Here the MDP surves as a way to solve the Reinforcement-learning problem. Reinforcement-learning is the problem of getting an agent to act in the world (environment) so as to maximize its rewards. The environment is modelled as a stochastic finite state machine with inputs and outputs [13]. Bellman's equation for defined MDP can be solved using value iteration or policy iteration algorithms.

We rewrite the equation (1.1) in the form of recursive iterations as follows

$$V^{n+1}(x_i) = \max_a \left(r_i + \gamma \sum_{j=1}^{n} P_{ij}^k V^n(x_j) \right). \tag{2.1}$$

The value iteration algorithm solves the given MDP by computing $V^n(x_i)$ for all i until converged. The algorithm has converged when

$$\max_i \left| V^{n+1}(x_i) - V^n(x_i) \right| < \epsilon.$$

The optimal policy is then defined as

$$\arg\max_a \left(R(x,a) + \gamma \sum_y P(x,a,y) V^*(y) \right). \tag{2.2}$$

Using policy iteration algorithm the $\pi(x_i)$ defines an action selected in the i'th state and is called a policy. Then the algorithm solves the given MDP computing $V^n(x_i)$ for all i using $\pi(x_i)$ and finding

$$\pi_k(x_i) = \arg\max_a \left(r_i + \gamma \sum_j P_{ij}^k V^n(x_j) \right). \tag{2.3}$$

The algorithm keeps computing until $\pi_k = \pi_{k+1}$. If this condition holds the optimal policy is found. The algorithms of value iteration and policy iteration require an explicit aproximation of R and P. Unlike those two algorithms, the Q-learning algorithm allows to find an optimal policy without explicitly aproximating R and P [8, 9]. The Q-learning algorithm estimates the optimal Q function

$$Q^*(x,a) = R(x,a) + \gamma \sum_y P(x,a,y) V^*(y). \tag{2.4}$$

Knowing the $Q^*(x,a)$ the optimal value function is found from $V^*(x) = \max_a Q^*(x,a)$. Given the experience at step t $\langle x_t, a_t, y_t, r_t \rangle$ and the current estimate $Q_t(x,a)$ the Q-learning updates to the next step

$$Q_{t+1}(x_t,a_t) := \big(1 - \alpha_t(x_t,a_t)\big) Q_t(x_t,a_t) + \alpha_t(x_t,a_t)\big(r_t + \gamma \max_a Q_t(y_t,a)\big). \tag{2.5}$$

Littman and Szepesvari [11] have shown that the Q-learning algorithm converges to the optimal Q function with probability 1 over $X \times A$.

3. Application of MDP in a virtual learning environment

Let us avoid the detailed structure of the multiagent system of the VLE [3] and define only four possible states of our abstract teacher agent. The first possible state we entitle Beginner, the second – Preintermediate, the third – Intermediate, and the fourth – Advanced, with the meaning that the agent knows the learner has beginner, preintermediate, intermediate, and advanced knowledge level in the defined Curricullum [3, 1, 2]. Actions that the agent could take in each state are enumerated as follows: the first – show topic material (STM), the second – give self-test with feedback (GST), the third – give evaluation test (GET). The rewards that the agent gets after transition to the corresponding state are 0, 5, 10 and 20. The probabilistic set of possible states of the agent is depicted in the Fig. 1. Our goal is to find an optimal policy for the teacher in the terms of reinforcement learning, which will show the actions of the teacher that are best to apply as a teaching method in order to maximize the future rewards and to reach the goal state as quick as possible. In this case the goal state is the fourth possible state of the teacher agent. Solving the given problem we apply the Q-learning algorithm. After iterative computation the matrix of Q-factors is computed.

This matrix has the number of rows equal to the number of the possible states of the agent and the number of columns equal to the number of actions available. In our case this is an 4×3 matrix. An action to take first in the learning phase of the algorithm

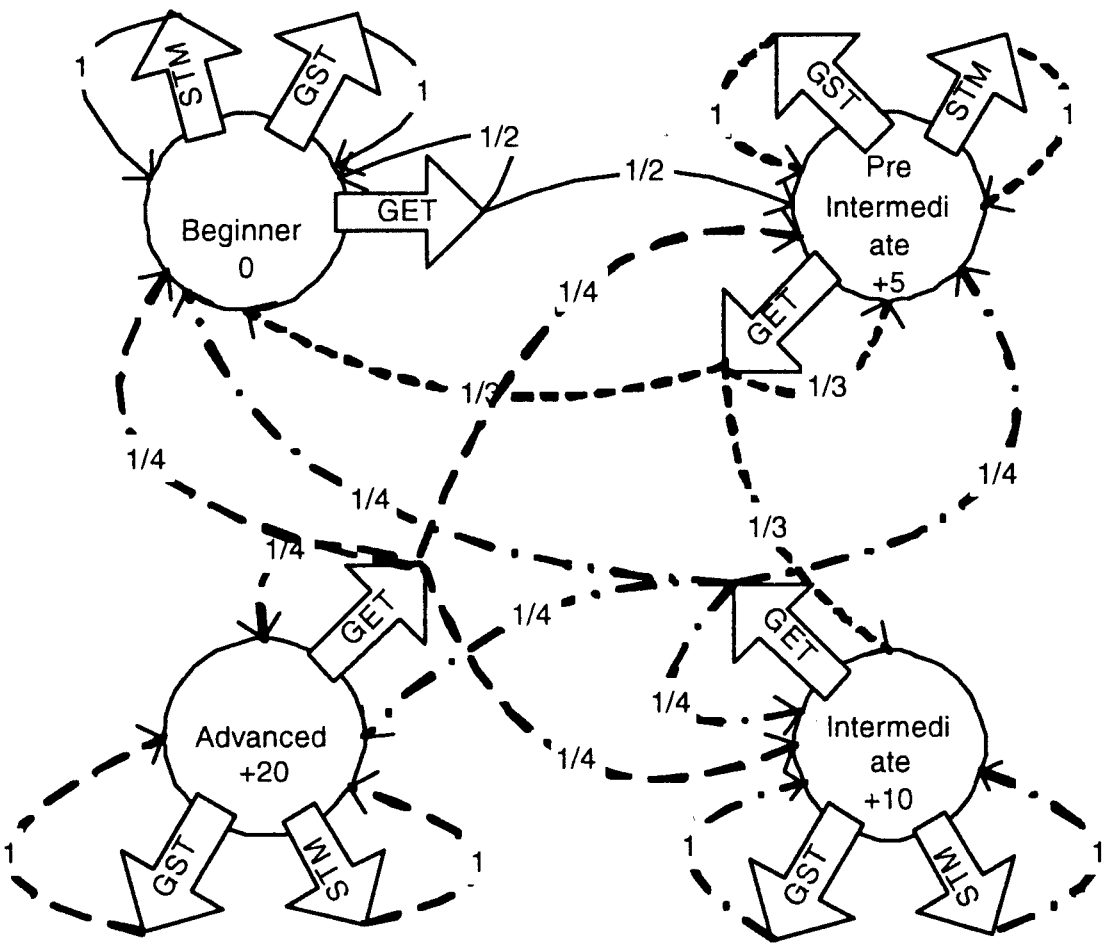

Fig. 1. The probabilistic set of possible states of the agent.

is defined stochastically by generating a random value. This means that values of Q-factors in the matrix Q may vary in the different runs of the algorithm. For example, after the rountine run of the algorithm we got the following matrix of Q-factors:

$$Q = \begin{pmatrix} 0.0003 & 0.0000 & 0.0335 \\ 0.8518 & 0.0295 & 3.0893 \\ 2.9585 & 1.9024 & 2.0360 \\ 11.5192 & 22.1915 & 1.5325 \end{pmatrix}.$$

From this matrix we find the optimal value function $V^*(x)$, which defines the optimal policy for our agent. Optimal value function selects the maximum Q-factors in each row of the given matrix. The maximum Q-factor has a meaning that corresponding action (the number of the column) is the best in the corresponding state (the number of the row). According to the given Q-matrix the optimal policy for the agent is defined by the mapping from states to actions $\pi(x)$: $X \to A$ as is shown bellow.

$\pi(x)$: $X \to A$

X	A
1 (Beginner)	3 (Give Evaluation Test)
2 (PreIntermediate)	3 (Give Evaluation Test)
3 (Intermediate)	1 (Show Topic Material)
4 (Advanced)	2 (Give Self-Test)

4. Conclusions

The investigations done have shown that the behavior of the tutor could be modelled applying the agent paradigm. For that purpose possible states, actions of an agent and rewards it receives should be introduced. The task of training the agent to apply the

optimal learning strategy could be formulated as the problem of RL and the Q-learning algorithm could be applied for finding such a policy.

References

1. D. Baziukaitė, A.A. Bielskis, O. Ramašauskas, Applying adaptive learning principles for the e-studies, *Liet. matem. rink.*, **42**(spec. nr.), 214–218 (2002).
2. D. Baziukaitė, Concept of adaptive based virtual learning environment, in: D. Rutkauskienė (ed.), *Proceedings of the International Conference TELDA'03*, Kaunas University of Technology, Kaunas (2003), pp. 63–66.
3. D. Baziukaitė, Multiagent system engineering application modeling curriculum setup sub-system, in: R. Krištolaitis (ed.), *9-oji magistrantų ir doktorantų konferencija*, Vytauto Didžiojo universitetas, Kaunas (2004), pp. 172–178.
4. D.L. Hobbs, A constructivist approach to web course design: a review of the literature, *International Journal on E-Learning*, April–June, 60–65 (2002).
5. Kinshuk, H. Hong, A. Patel, Adaptivity through the use of mobile agents in web-based student modelling, *International Journal on E-Learning*, July–September, 55–64 (2002).
6. G. Weber, M. Specht, User modeling and adaptive navigation support in www-based tutoring systems, in: A. Jameson, C. Paris, C. Tasso (Eds.), *User Modeling: Proceedings of the Sixth International Conference, UM97*, Springer, Wien, New York (1997), pp. 289–300.
7. R.S. Sutton, A.G. Barto, *Reinforcement Learning: an Introduction*, MIT Press, Cambridge (1998).
8. L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, *A Journal of Artificial Intelligence Research*, **4**, 237–285 (1996).
9. A. Gosavi, *Tutorial for Reinforcement Learning* (2004). `http://www.eng.buffalo.edu/ agosavi/tutorial.pdf`
10. A. Gosavi, Reinforcement learning for long-run average cost, *European Journal of Operational Research*, **155**, 654–674 (2004).
11. M.L. Littman, C. Szepesvari, A generalized reinforcement-learning model: convergence and applications, in: *Proceedings of the Thirteenth International Conference on Machine Learning* (1996), pp. 310–318.
12. A. Iglesias, P. Martinez, F. Fernandez, An experience applying reinforcment learning in a web-based adaptive and intelligent educational system, *Informatics in Education*, **2**(2), 223–240 (2003).
13. K. Murphy, *Markov Decision Process Toolbox for Matlab* (2002). `http://www.ai.mit.edu/ murphyk/Software/MDP/mdp.html`

REZIUMĖ

D. Baziukaitė. Intelektualinės virtualios mokymo(si) sistemos: Markovo sprendimo priėmimo proceso taikymas

Straipsnyje aptariamos baigtinio Markovo sprendimo priėmimo proceso taikymo prielaidos virtualioms mokymosi sistemoms intelektualizuoti parodant, jog virtualioje mokymosi sistemoje veikiančių agentų būsenų aibė, agentų priimami sprendimai ir perėjimai iš vienos būsenos į kitą gali būti modeliuojami taikant Markovo sprendimo priėmimo procesų ir sustiprinto mokymosi (Reinforcement-Learning) teorijos rezultatus. Intelektuali mokymosi sistema suprantama kaip gebanti parinkti optimalų mokymosi kelią besimokančiajam ir koreguoti savo veiksmus atsižvelgdama į laikui bėgant įgytą patyrimą. Sustiprinto mokymosi problemai, optimalios vertės funkcijos $V^*(x)$ radimui, spręsti galima taikyti įvairius algoritmus. Vienas jų – tai Q-mokymosi (Q-learning) algoritmas, leidžiantis surasti optimalią $Q^*(x, a)$ funkciją. Stochastinės aproksimacijos teoremos sąlygų įvykdymas Q-mokymosi algoritme užtikrina algoritmo konvergavimą į optimalią $Q^*(x, a)$, o tai reiškia ir į optimalią vertės funkciją $V^*(x)$.