The Sixteenth International Symposium on Artificial Life and Robotics 2011 (AROB 16th '11),
B-Con Plaza, Beppu,Oita, Japan, January 27-29, 2011

Temporal difference approach in linearly-solvable Markov decision problems

Burdelis, M.A.P. and Ikeda, K.

Nara Institute of Science and Technology, Grad. School of Information Science,
A606, 8916-5 Takayama, Ikoma, NARA, 630-0192, Japan
(Tel : 81-743-72-5984; Fax : 81-743-72-5989)
(mauricio-b@is.naist.jp)

Abstract: Todorov has recently introduced a class of linearly-solvable MDPs (LSMDPs) which greatly simplifies reinforcement learning. By attending some specific conditions, the problem of choosing optimal actions (sequential decision making) can become linear, and then be solved in closed-form. A similar method to temporal difference learning (TD learning) for this class of MDPs has also been introduced, and is called Z-learning. In this work we present the results of simulations using Z-learning to solve a navigation problem of a virtual agent in a grid world, in which physical properties of the system (Newtonian mechanics) were introduced in the MDP model in the definition of the passive dynamics probability distribution, which is crucial in the theory. Those MDPs were solved both in closed-form and using Z-learning. In all experiments, the approximation errors of Z-learning consistently decreased with the increase in the number of simulation steps.

Keywords: Linear Bellman Equation; Reinforcement Learning.

I. INTRODUCTION

The Reinforcement learning (RL) approach to Machine Learning is a technique to learn how to make decisions in order to achieve a desired goal. In RL, the model does not include the presence of a supervisor, and the agent must learn by trial and error, interacting with the environment and observing a reward (or cost) signal [1]. Examples of possible applications of RL include: playing board games like chess, checkers or "tic-tac-toe"; and a nervous system generating muscle activations to maximize movement performance [2].

RL problems are usually defined on a discrete-time Markov decision process (MDP) with stochastic dynamics. Recent work [3] has presented a class of linearly-solvable MDPs, which greatly simplify the solution of reinforcement learning problems. Z-learning is a temporal-difference approach to the RL problem, which takes advantage of this class of MDPs, and presents faster convergence than traditional RL methods (e.g. Q-learning) [3] [2]. The present work has the motivation of including Newtonian mechanics effects (inertia and collisions) in the passive dynamics probability distribution of the linearly-solvable MDP used for Z-learning.

II. LINEARLY-SOLVABLE MARKOV DECISION PROCESSES (LSMDP)

Let us denote the state of the environment in a discrete time instant t as x_t. If this state signal retains all relevant information for the decision, it is said to have the "Markov property", and

$$\Pr(x_{t+1} \mid x_t) = \Pr(x_{t+1} \mid x_t, x_{t-1} \ldots) . \tag{1}$$

This means that the history of past states has no influence on the probability of the next state. A reinforcement learning task that satisfies the Markov property is called a Markov decision process (MDP) [1] [4].

A particular finite MDP is defined by: a set of possible states X, a set of possible actions U, state transition probabilities $p(x_{t+1}|x_t, u_t)$ (which mean the probability that the next state is state x_{t+1} when the current state is state x_t and action u_t is taken), and immediate cost $l(x_{t+1}|x_t, u_t)$ for being at state x_t, taking action u_t and having a transition to state x_{t+1}.

The reinforcement learning agent's sole objective is to minimize the total accumulated cost it receives in the long run [1] [2]. The "cost-to-go" function (or "value" function) of a state (denoted $v(x)$) is defined as the expected total cost the agent accumulates starting from that state, and following the optimal policy thereafter.

The cost-to-go (or "value") function is the only solution to its Bellman equation [1] (refer to section III for details). This solution can be obtained by using Dynamic Programming (DP), but this can be time-consuming due to the explosion of unknown variables.

Recent work [3] has introduced a class of MDPs which greatly simplifies reinforcement learning. By attending some specific conditions, the Bellman equation of the MDP becomes linear, and its solution can be obtained in closed-form. The conditions are as follows [2]:

In the original MDP framework, the agent specifies discrete actions $u \in U$. The probabilities of state transitions starting from the current state $x \in X$ depend on the action u taken at that state. In other words, the probability of transition from a state $x \in X$ to a state $x' \in X$ is given by $p(x'|x, u)$.

In LSMDPs, however, the agent can specify the transition probabilities directly. In other words, there are no discrete actions u nor the set of actions U. Instead, the agent can directly specify the probability of transition from the current state $x \in X$ to any possible future state $x' \in X$, without the existence of discrete actions. These probabilities will be represented by using the letter u and therefore $p(x'|x, u)$ becomes $u(x'|x)$ [2].

Another necessary condition is the definition of a probability distribution called "passive dynamics", denoted $p_d(x'|x)$, which corresponds to the behavior of the system in the absence of controls [2].

The total cost incurred in a state transition will be defined as follows [2]:

$$l(x,u) = q(x) + KL\big(u(\cdot\,|\,x) \,\|\, p_d(\cdot\,|\,x)\big) \qquad (2)$$

Where $q(x)$ is called "state-cost" (which depends only on the state, therefore representing how undesirable a state is) and

$$\mathrm{KL}\big(u(\cdot\,|\,x) \,\|\, p(\cdot\,|\,x)\big) \equiv E_{x'\sim u(\cdot|x)}\left[\log \frac{u(x'|\,x)}{p_d(x'|\,x)}\right] \qquad (3)$$

is the Kullback–Leibler (KL) divergence between the controlled state transition distribution and the passive dynamics. This measures how "different" these distributions are from one another, and is called "action cost" [2].

The last condition is that $u(x'|x)=0$ when $p_d(x'|x)=0$, in order to keep the KL divergence well-defined and avoid impossible state transitions [2].

III. Z-LEARNING

When the passive dynamics distribution is known, as well as all the states and their respective state costs, then the problem of obtaining the cost-to-go function $v(x)$ can be solved using dynamic programming and the Bellman equation. The Bellman equation expresses the relationship between the cost-to-go of state and the expected cost-to-go of the next state:

$$v(x) = \min_u \left\{ l(x,u) + E_{x'\sim p(\cdot|x,u)}[v(x')] \right\} \qquad (4)$$

where

$$E_{x'\sim p(\cdot|x,u)}[v(x')] \equiv \sum_{x'} p(x'|\,x,u)v(x')\,. \qquad (5)$$

In LSMDPs, the Bellman equation can be expressed as [2]:

$$z(x) = e^{(-q(x))} \sum_{x'} p(x'|\,x) z(x')\,, \qquad (6)$$

where

$$z(x) \equiv e^{(-v(x))} \qquad (7)$$

is called the "desirability function" of a state. This Bellman equation (6) is linear in z.

Equation (6) can be written in vector notation, by enumerating the states from 1 to n, representing $z(x)$ and $q(x)$ as column vectors $\mathbf{z}$ and $\mathbf{q}$, and representing $p(x'|x)$ as a matrix $\mathbf{P}$ (where the row-index corresponds to x and the column-index corresponds to x') [2]. By partitioning $\mathbf{z}$, $\mathbf{q}$ and $\mathbf{P}$ according to terminal and non-terminal states (using index N for non-terminal and T for terminal), equation (6) becomes [2]:

$$(\mathrm{diag}\ (\exp(q_N)) - \mathbf{P}_{NN})\mathbf{z}_N = \mathbf{P}_{NT} \exp(-q_T) \qquad (8)$$

where "diag" transforms vectors into diagonal matrices. By acknowledging that $v(x) = q(x)$ at terminal states, the unknown $\mathbf{z}_N$ (vector of desirabilities at the non-terminal states) can be calculated by using matrix factorization or an iterative linear solver [2].

When the state costs and passive dynamics are not known, then simulations must be made in order to solve the problem of estimating the cost-to-go function by using reinforcement learning.

Z-learning is a temporal-difference-like method which takes advantage of the linear class of MDPs to achieve faster convergence then traditional reinforcement learning methods [2].

As in traditional TD learning methods, initial estimates are constantly updated until convergence is obtained. The Z-learning update formula is as follows [2]:

$$z_{upd}(x_t) \leftarrow (1-\eta_t) z_{cur}(x_t) + \eta_t \exp(-q_t) z_{cur}(x_{t+1}) \qquad (9)$$

Where $z_{upd}(x_t)$ is the updated estimate of $z(x_t)$, $z_{cur}(x_t)$ is the current estimate of $z(x_t)$, $z_{cur}(x_{t+1})$ is the current estimate of $z(x_{t+1})$, q_t is the state cost of state x_t, and η_t is a learning rate which decreases over time.

IV. MODELING NEWTONIAN MECHANICS EFFECTS

In the present work two Newtonian mechanics effects were modeled in the passive dynamics probability distribution (which is usually defined simply as "random walk"): inertia and collisions. These Newtonian mechanics effects were modeled and simulated in a 2-dimensional "grid-world" (10x10 size, with obstacles - Fig.1). The goal of the agent is to find the goal position, while avoiding the obstacles on the path.

Initially, a few difficulties have emerged from modeling the state signal in the grid world as the position of the learning agent. In order to model inertia, it is necessary to know the current and the previous position of the agent. If the state signal contains only the information on the current position, then it is necessary to know the current and the previous state in order to model the passive dynamics, and this would violate the "Markov property" (1).

Also, in order to model collisions with obstacles and walls, the agent would originally need to be able to take discrete actions of movement in the direction of the wall or obstacle, and this would violate one of the necessary conditions to have a LSMDP.

In order to overcome these two difficulties, the state signal was modeled to include information of position pairs: the current and the previous position.

A simple model of inertia was created by assigning a pre-defined "highest probability" value (denoted *hp*) to the passive dynamics probability of the next position in the current trajectory of the agent. The remaining adjacent non-obstacle positions equally share the probability *(1-hp)* (Fig. 2).

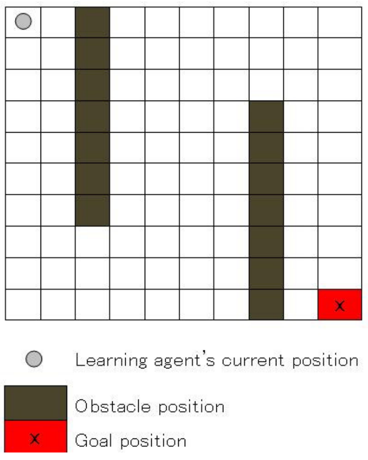

Fig.1. A two-dimensional "grid world" (size 10x10)

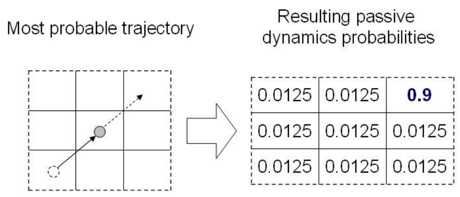

Fig.2. Modeling inertia (with hp = 0.9)

Two types of walls and obstacles were considered in order to model collisions: reflexive walls and obstacles (which reflect the normal components of impacts); and absorptive walls and obstacles (which absorb the normal components of impacts). The passive dynamics probability distribution is updated based on the current position and the previous position of the agent. When the agent's position is adjacent to a wall or obstacle, and the most probable trajectory (driven by inertia) is in the direction of the wall or obstacle, then the position that receives the highest probability value *hp* is the most likely position after a collision with this wall or obstacle, and the remaining possible positions share the remaining *(1-hp)*, as shown in Fig.3

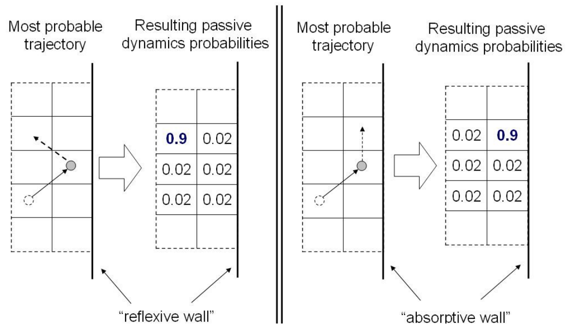

Fig.3. Collisions with a reflexive and an absorptive wall (with hp = 0.9)

V. COMPUTATIONAL EXPERIMENTS

The grid world shown in Fig.1 was solved analytically (using equation (8) [2]) and also using Z-learning. The state cost of every state (except for the goal state) is *1* (in order to penalize the agent for producing long trajectories) and *0* only at the goal state. Both reflexive and absorptive walls and obstacles were considered. In Z-learning, three different policies were used: policy equal to the passive dynamics; random walk policy; and ε-greedy policy (with $\varepsilon=0.2$). When the applied policy differs from the passive dynamics distribution, importance sampling is required [2]: the last term in equation (9) needs to be multiplied by $p_d(x_{t+1}|x_t)/\hat{u}(x_{t+1}|x_t)$, where $\hat{u}$ is the applied policy – in this case it is necessary to know the passive dynamics distribution p_d.

In order to measure the quality of the approximation of the cost-to-go function estimates produced by Z-learning, the approximation error (comparing the estimates with the correct values obtained in closed-form) was calculated using the formula:

$$Error = \sum_{i=1}^{N} \left| v(x_i) - v^*(x_i) \right| \Big/ \sum_{i=1}^{N} v^*(x_i) \qquad (10)$$

where x_i represents the i^{th} state; N is the total number of states; $v(x)$ is the current approximation of the cost-to-go function (obtained by Z-learning) and $v^*(x)$ is the optimal cost-to-go function obtained analytically.

The results of the simulations can be seen in Fig.5 and Fig.6. In all experiments the learning rate η_t decays obeying the formula $\eta_t=c/(c+t)$ where t is the time step and c is a constant which was different for each case. The results for both models (the model with absorptive walls and obstacles; and the model with reflexive walls and obstacles) are very similar, indicating that the framework is considerably robust. When sampling from the passive dynamics, the error takes more steps to converge, because the inclusion of inertia in the passive dynamics makes this distribution less exploratory than random walk. The other two policies (random walk and ε-greedy) presented faster convergence, but their use requires access to p_d. In all cases the Z-learning approximation errors consistently decrease as the number of simulation steps increases.

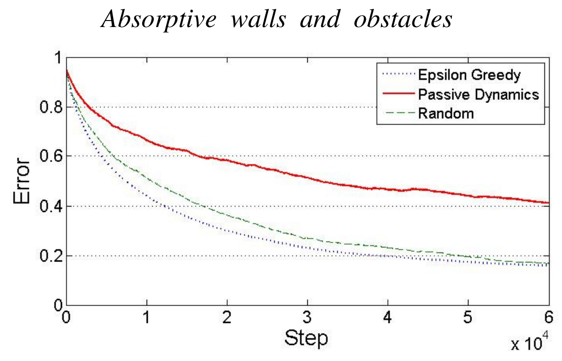

Fig.5. Z-learning approximation errors (absorptive walls and obstacles)

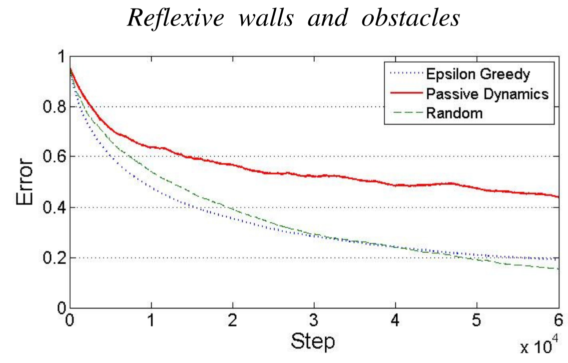

Fig.6. Z-learning approximation errors (reflexive walls and obstacles)

VI. CONCLUSION

The flexible framework of linearly-solvable MDPs allowed for simple models of Newtonian mechanics effects to be simulated in the passive dynamics probability distribution. The cost-to-go function of the resulting model was obtained both in closed-form and using Z-learning. The Z-learning algorithm was able to approximate the correct cost-to-go function, presenting errors which consistently decreased with the increase in the number of simulation steps.

REFERENCES

[1] Sutton R S, Barto A G (1998). Reinforcement learning: An introduction. MIT Press.
[2] Todorov E. (2009) Efficient computation of optimal actions. Proc Natl Acad Sci USA 106:11478–11483.
[3] Todorov E (2006) Linearly-solvable Markov decision problems. Adv Neural Informatin Proc Syst 19:1369–1376.
[4] Iwata K, Ikeda K, Sakai H (2006). The asymptotic equipartition property in reinforcement learning and its relation to return maximization. Neural Networks 19(1), 62–75.