A proposition of addition and integration of q-values in Q-Learning

Kathy Thi Aung[1], Takayasu Fuchida[2]

[1,2]Kagoshima University, Kagoshima 890-0065, Kohrimoto 1-21-40, Japan
[1,2]Graduate School of Science and Engineering, Department of System Information Science
(Tel: 81-99-285-3408, Fax: 81-99-285-8464)

[1]kathythiaung@gmail.com , [2]fuchida@ibe.kagoshima-u.ac.jp

Abstract: This article presents a method of addition and integration of q-values to reduce the number of states and memory usage based on Q-learning algorithm in continuous state space using the concept of Voronoi space division. It also aims to show the improvement of learning efficiency when it is compared to the existing method, such as lattice. We constructed an experimental model to examine these scenarios. This model is based on continuous state and discrete action of feeder mouse. The results indicate that the proposed method greatly improve than normal Q-learning. As a method of space division, we used Voronoi diagram that is a general space division method. However, Voronoi diagram has a lot of flexibility therefore it becomes a problem to decide the position of q-values. Therefore, we presented the addition method in order to realize the position of q-values using LBG algorithm though there are many methods for adaptive vector grouping. In addition, we integrate the q-values which have the same action selections using Delaunay tessellation technique to find the nearest q-values.

Keywords: Q-Learning, Voronoi Diagram, LBG algorithm, Delaunay Tessellation technique

1 Introduction

There are several kinds of learning methods but reinforcement learning (RL) is the most suitable method in machine learning that deals with the decision to take an action using an agent at discrete time steps and expected that would be useful anywhere in the future [1]. RL methods attempt to improve the agent's decision-making policy over the time. The agent's goal is to get as much reward as it can over the long run.

In this paper, we present an experimental study in continuous state space based on Q-learning algorithm with addition and integration of q-values method using Delaunay tessellation technique, addition method of q-values using LBG algorithm, and normal lattice arrangement. It also aims to show the improvement of learning efficiency using the proposed method when it is compared to the existing method, such as lattice.

We constructed a computational model to examine these scenarios. In the model, an agent aims to get as much reward as possible during a specific period, and observes the angle (θ) and the distance (d) to reward-area. If the agent reaches to the reward-area, the agent gets a reward of +1 and the position of agent is randomly changed.

2 Methods and procedures

2.1 Q-Learning algorithm

There are several ways to implement the learning process. However, Q-learning algorithm due to Watkins [2] is a policy for estimating the optimal state-action value, and one of the most fundamental in RL. Q-learning can also apply in many practical applications based on the idea of expected future reinforcements, and can be estimated by the function of each action in each state. However, it works only for discrete state space, and difficult to handle in continuous state space because of curse of dimensionality problem therefore it needs to discretize the state space into a lot of smaller discrete regions when we treat such continuous cases.

Curse of dimensionality means if the number of state and action variables increase, the size of the Q-Table used to store Q-values grows exponentially and the learning speed decrease suddenly.

In general Q-learning, every action and state pair have their own Q-value denoted by $Q(s_t,a_t)$, and it stores in a table called Q-Table. It looks like a square lattice in 2-dimesions. These Q-values are initialized to small random numbers and gradually change toward the optimal values through learning. The Q-value is updated in taking the maximum Q-value of next state using this following equation.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t))$$

In this equation, a state in time t is s_t , Q-value of action a_t in the state s_t is $Q(s_t,a_t)$, α is the learning rate, γ is the discount rate, and r is a reward. Furthermore, *max* $Q(s_{t+1},a)$ shows the maximum Q-value of next state in time t+1.

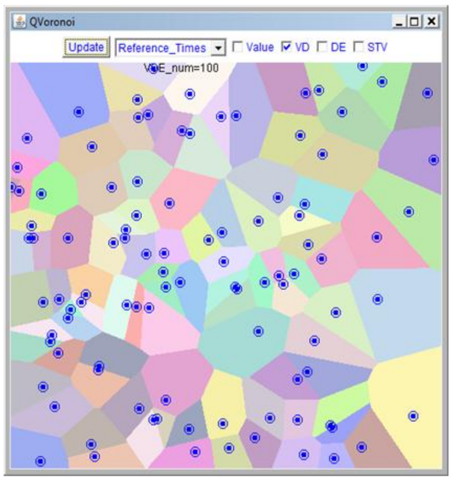

Fig.1. Voronoi diagram in a random set of point

2.2 Voronoi diagram

Voronoi diagram is used in our approach to partitions space into cells or a number of regions, where each region consists of all points that are closer to one site than to any other. It can also be used to perform nearest-neighbor searches. As a simple illustration, show in figure 1. Voronoi regions are bounded by line segments and Voronoi edge is a bisector of two sites whose regions are adjacent.

2.3 Addition of q-values

In the first stage of addition process, we put the temporary points into lattice structure on state space firstly. When the learning process is started, we save the state transition vector (STV) that enters from the position of the agent to the reward area on action space. If we get 1000 STV or more, we quantize or group those STV by continuously taking the same action using LBG algorithm (section 2.3.1). Furthermore, we seek representative vectors and describe STV groups as representative vectors.

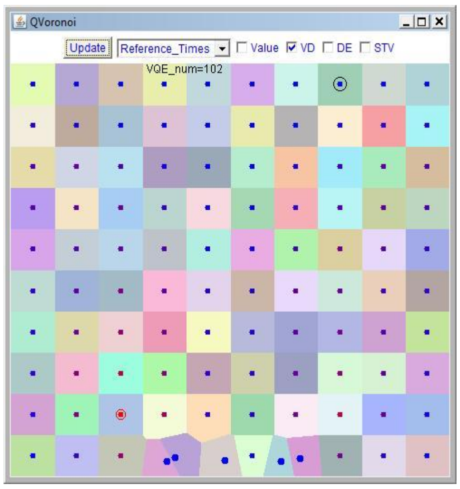

Fig.2. First-stage of q-values addition

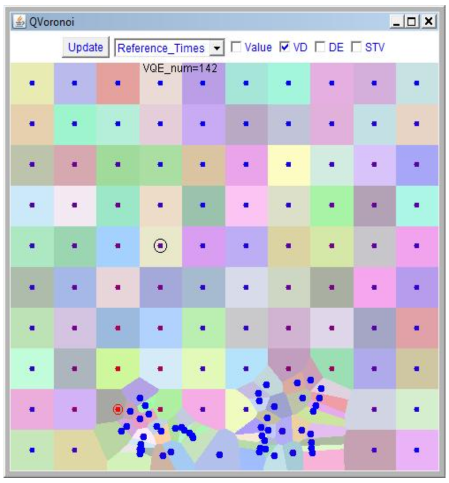

Fig.3. An image of q-values addition

Additionally, generate the new point at the place of each representative vectors. Moreover, delete the temporary points in which new points are added. A first-stage formation of addition is illustrated in figure 2.

In the second stage of addition, we save STV again that come from temporary points to new points of the first stage output. Nevertheless, we do not take STV that come from new points to new points. In addition, generate the new points at the position of the representative vectors and repeat the above process. In algorithm, a new point is put on the position of a suitable multiple at the direction of representative state transition vectors but we do not judge yet whether that position is really appropriate location or not. Figure 3 shows the image of addition of q-values.

2.3.1 LBG algorithm

We used LBG (Linde-Buzo-Gray) algorithm for vector quantization. It is like a clustering algorithm which takes a set of input vectors as input and generates a representative subset of vectors with a quantum vector. The modification of adaptive vector quantization method was introduced

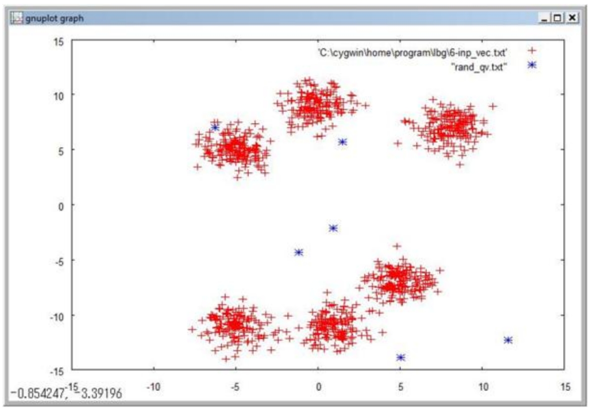

Fig.4. Initial state of vector quantization

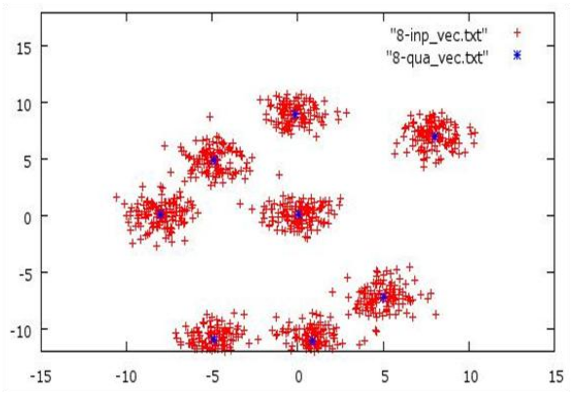

Fig.5. Illustration of the LBG vector quantization

enhanced LBG (Patane & Russo, 2001) [3], and adaptive incremental LBG (Shen & Hasegawa, 2006) [4]. LBG algorithm is as

1. Collect input vectors.
2. Initiate quantum vector at random.
3. Make group that input vectors belongs to the cluster of nearest quantum vector that yields minimum distance.
4. Shift quantum vector to the gravity center of the group dividing the own coordinates by the number of input vectors.
5. Move quantum vector in random direction and repeat the process in several times until the amount of quantum vector is less than a threshold value of 0.1.

The convergence of LBG algorithm depends on the initial quantum vector and the threshold in implementation. Figure 4 describes an initial state of vector quantization algorithm. The input vectors are marked with red and quantum vector are marked with blue. Figure 5 obtained by using the above mentioned algorithm.

2.4 Integration of q-values

There are five conditions to integrate q-values.

(1) both two points must be new added points.

(2) Those two adjoining points must take the same optimal action (Figure 6).

(3) Both two optimal actions have not changed over the number of 10,000 action times in 500,000 integrated timing of new points.

(4) Both two new points are not already used for integration.

(5) Start the integration process after the number of q-values is added to 300 as a restriction of integration theory and reduces to 169.

A new point is added to the center of adjoining point, and those two adjoining points are deleted. The key point of our integration method is to integrate same actions.

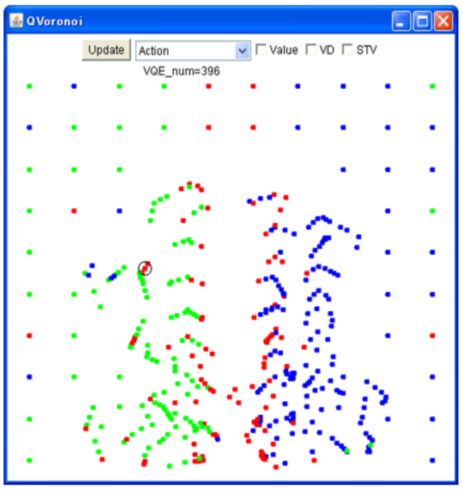

Fig.6. Type of actions on state space for integration

2.4.1 Delaunay tessellation algorithm

We applied a Delaunay tessellation (DT) technique to integrate the closest q-values. Delaunay tessellation is another fundamental computational geometry structure and dual tessellation of Voronoi diagram. DT is the straight-line dual of the Voronoi diagram obtained by joining all pairs of points belongs to the set. In our algorithm, we assume that the number of points are three or more but finite.

All triangles of the Delaunay triangulation are obtained by connecting the adjoining points. Then, we draw a hypercube rectangle containing all points that adjacent to nearest point and generate a random point inside of inclusive rectangle. After that, it finds the connection of nearest point and second-nearest point by measuring the distance from that random point. Finally, two nearest points with high mark are connected. An example of java applet animation of Delaunay tessellation is shown in figure 7.

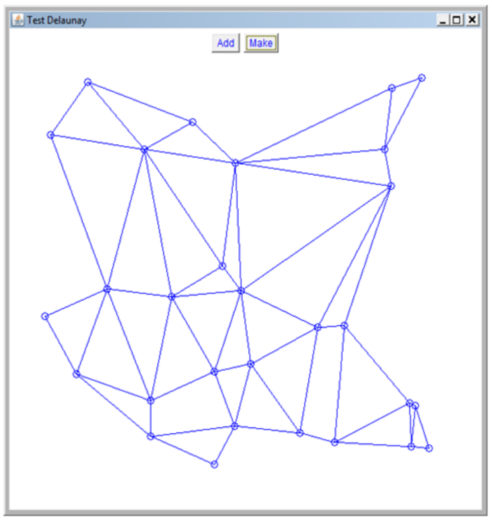

Fig.7. Java applet animation of Delaunay tessellation

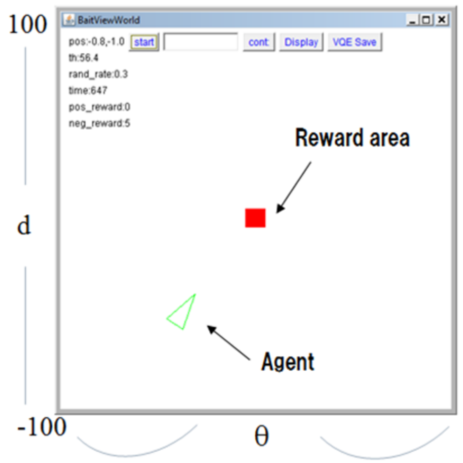

Fig.8. Bait View World experimental model

3 Experiments

In the simulation, an agent is supposed to head toward the reward-are. We call this model "bait view world", also called "non-coincidence of state space and action space model", shown in figure 8. An agent gets a reward when it enters the reward-area and the agent's position change randomly on this bait view world. The objective of agent is to get as much reward as possible during a specific period. This model is based on continuous state and discrete action of feeder mouse.

The agent observes the angle and distance to reward-area as shown in figure 9 and these 2 input values construct the 2-dimensional state space. The agent has 3-types of control actions selection; 1) go straight ahead, left rotation and right rotation. Each action is displayed by color "red", "green" and "blue" (Fig 6). The agent normally selects the next action which has a maximum Q-value but sometimes selects the next action at random.

We conducted the experiments with 100,000 continuous learning times make one "episode" and executed 160 episodes in one "experiment". We did 10 trials for each episode by changing the random initial seed and took an average. The learning rate **α** was set to 0.1 and discount rate **γ** was set to 0.9.

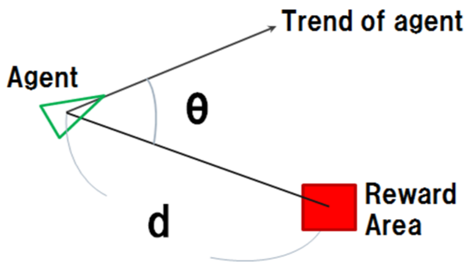

Fig.9. 2-input values for experimental model

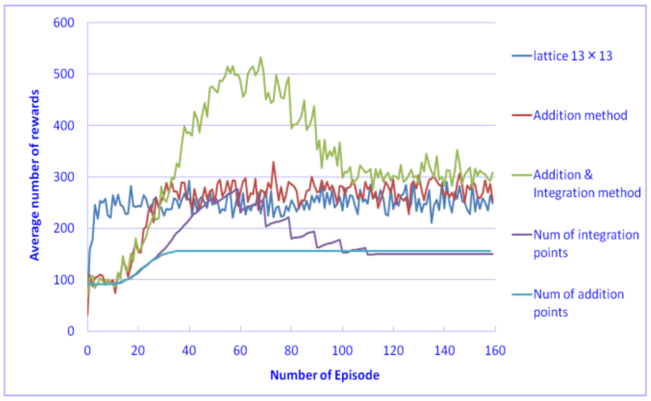

Fig.10. Experimental results of 3-methods

3.1 Experimental results

We examined the performance of the following several different methods in a stationary situation of reward-area through the computer simulations. These are 1) lattice arrangement of normal Q-learning on discrete state space, 2) implementation of addition method using LBG algorithm, and 3) implementation of integration method using Delaunay tessellation technique on continuous state space. Figure 10 shows the result of these three experiments.

4 Conclusions

We presented the addition and integration method of q-values on continuous state space in order to realize the position of Voronoi points and to reduce the number of states and to speed up the learning efficiency. Moreover, we investigated the performance and efficiency of our proposed methods using Bait View World experimental model. As a result of experiments, the number of state decreases greatly and our proposed methods give good results in simulation. However, we occurs the over-integration problem though the number of states has decreased and the learning speed is suddenly decreased at the halfway. Therefore, we need to consider about this problem and it is need to able to change the integrated timing adaptively. Furthermore, we applied Voronoi diagram, LBG algorithm and Delaunay Tessellation technique in this study.

References

[1] Sutton RS, Barto AG (1998), Reinforcement learning an introduction, MIT Press, Cambridge.
[2] Watkins CJCH, Dayan P. Technical notes: Q-learning. Machine Learning 1992;8:279-292.
[3] G.Patane and M.Russo, The enhanced LBG algorithm, In proceedings of Neural Networks, 2001, pp.1219-1237.
[4] F.Shen, O.Hasegawa, An adaptive incremental LBG for vector quantization, Neural Networks 19 (2006) 694-704.