Fuzzy-Q learning

서일홍*, 김재현*, 오상록**

* 한양대학교 전자공학과 Intelligent & Robotics Lab.
** 한국과학기술연구원 정보전자부

요 약

Robotic System 은 미리 계획되지않은 상황에 대처하는 능력이 있어야만 자율적인 능력을 요구하는 곳에 적용될 수 있다. 이를 위하여 일련의 상황에 대해 적절한 일련의 행위들을 학습할 수 있는 학습 방법이 요구된다. Q-learning 은 불연속 상태 공간과 행위 공간을 정의하고 현재상태로부터 목표상태에 도달하기 위한 최적의 행위 집합을 구하기 위한 통계적인 해결책으로 제안되었던 바, 이를 실질적인 연속상태공간에 적용하기 위해서는 너무 많은 기억 공간을 요구하며, 또한, 학습되는 출력(행위)공간이 불연속이기 때문에 부드럽지 못한 행위를 발생시키는 단점을 항상 갖게 되었다. 본 논문에서는 기존의 Q-learning 의 단점을 극복하기 위해 Fuzzy 개념을 도입하여 연속된 상태와 연속된 행위간의 적절한 대응을 학습하는 Fuzzy Q-learning algorithm 을 제안하고자 한다.

1. 서론

최근 들어, control, planning, decision making 등을 실시간으로 처리하고자 하는 intelligent system 에 관한 연구가 많은 관심을 끌게 되었다. 이러한 여러 연구들에서 Reinforcement learning 과 같은 새로운 algorithm 의 개발[1,4]과, dynamic programming 과 같은 고전적인 algorithm 의 개선이 동시에 이루어져 왔다. 특히, Reinforcement learning 은 환경에 대한 충분한 지식 없이 주어진 환경에 적절한 행위를 학습할 수 있는 효과적인 방법으로 알려져 있다. 이러한 Reinforcement learning 중 Sutton 의 Temporal Difference(TD) method 는 Tesauro 에 의해 개선되어 backgammon-playing program 에 적용되었다. 특히, TD algorithm 과 밀접한 관련이 있는 Wakin 의 Q-learning algorithm[2]은 최근 들어 집중적으로 연구되어 Lin 은 이와 backpropagation 을 이용하여 Robot Control 에 적용하였다. 한편, Watkin 과 Dayan 은 Q-learning 의 convergence 에 관한 결과를 발표하였으며 Moore 와 Atkeson 은 Q-learning 의 수렴 속도를 높이기 위한 방법으로 prioritized sweeping technique 을 개발하였다. 이러한 대부분의 기존 연구들은 불연속 상태 공간(discrete state space)와 불연속 행위 집합(discrete action set)을 기본으로 진행되어 왔다. 그러나, 실질적인 Robotic application 에서 이러한 algorithm 을 적용하기에는 여러 가지 한계를 극복하여야만 한다. (1)먼저, 너무 많은 memory 를 필요로 하기 때문에 적용상 많은 어려움이 따르게 된다. (2)또한, 출력(행위)이 불연속이라는 제한을 갖게 된다. 예로써 Q-learning 의 경우 각 상태에서 각 행위의 가치을 기록하기 위한 Q-table 이 필요한데, 2-D 상태 공간(state space)에서 각 상태축이 100 개의 분해능(resolution)을 갖고, 100 개의 가능한 행위을 가정하더라도 이를 위해 필요한 Q-table 의 크기는 1000000 개 이상의 기억공간이 필요하게 된다. 따라서, 이러한 기억공간의 문제를 극복하면서도 기존의 Q-learning 과 대등한 성능(즉, iteration time, learning rate)을 갖는 새로운 알고리즘이 요구된다. 또한 이러한 기억 공간이 확보되었다 할지라도 Q-learning 은 불연속된 출력을 학습하는 방법이므로 항상 해당 분해능에 반비례하여 최적의 행위에 대한 학습 error 가 존재하게 된다.

본 논문에서는 기존의 Q-learning 의 단점을 극복하기 위해 먼저 각 상태에 대해 1 개의 행위에 대해서만 최고치를 갖는 Q-table 의 모델을 가정하고 Fuzzy 개념이 도입하여 연속된 상태와 연속된 행위간의 적절한 대응을 학습하는 Fuzzy Q-learning algorithm 을 제안하고자 한다. 즉, 상태 공간이 커짐에 따라 기존의 Q-table 을 기반으로 한 접근 방법은 많은 기억 공간을 요구할 뿐만 아니라 이에 비례하여 검색 시간도 증가한다는 것과 실질적인 행위가 policy table 에 근거한다는 것에 착안하여, Q-table 에서 policy table 을 얻는 기존의 방법을 연속된 Q-table 내의 최대 Q-value 을 추적하는 새로운 policy 개선 방법으로 대체하였다. 앞으로의 본문에서 우선 Markovian decision process 에 근거한 Watkin 의 Q-learning 을 2 장에서 설명하고, 이를 응용한 Fuzzy Q-learning 을 3 장에서 소개하고자 한다.

2. Q-learning algorithm

2.1 Q-learning algorithm 소개

강화 학습은 최근 들어 아무런 선험 지식 없이도 학습할 수 있고, 높은 reactive 특성과 적응성이 뛰어난 행위를 생성하는 robot learning method로서 많은 관심을 끌고있다. 그림1은 기본적인 로보트와 환경과의 상호작용 model을 나타낸다. 먼저 로보트는 환경에 대한 현재상태를 감지하여 적절한 행위를 선택한다. 상태와 행위에 근거하여 환경은 새로운 상태로 전이되고 행위에 대한 보답(reward)을 발생시키며, 이를 robot에게 되돌려 준다. 이러한 상호작용을 통해 robot은 현재상태에서 적절한 행위을 배우게 된다.

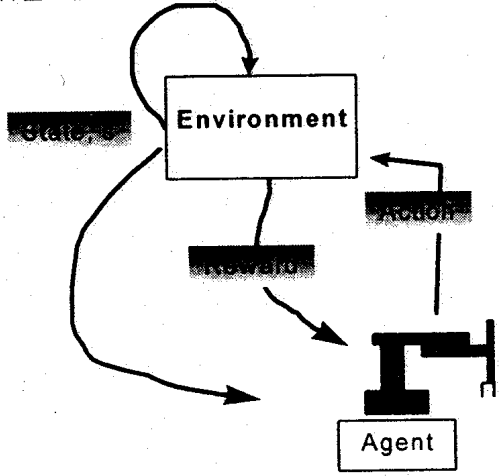


그림 1 로보트와 외부 환경과의 상호 작용 모델

이러한 관계를 정리하여 이론화한 Q-learning 알고리즘은 다음과 같다. 우선 로보트는 서로 다른

유한의 상태들의 집합 S를 감지할 수 있고, 유한의 행위들의 집합 A를 행할 수 있으며, 또한 외부 환경은 목표상태(goal state)로의 수렴성이 보장되는 Markov process로써 모델링할 수 있다고 가정하자. 여기서 $T(s, a, s')$를 현재 상태 s에 있는 로보트가 행위 a를 발할 때 현재상태 s와 행위 a 사이의 관계로부터 현재상태 s가 다음상태 s'로 변하게 될 상태전이확률 이라 하며, 이 때, 각 상태와 행위에 대한 보답 (reward)을 $r(s, a)$라 하자. 일반적인 강화학습 (reinforcement learning)은 시간 축에 걸쳐 얻어지는 보답(reward)의 합을 극대화하는 행위들의 책략(policy)을 찾는 것으로 정의 된다. 이러한 f는 S로부터 A로의 단순한 대입을 의미한다. 여기서 보답들의 합을 다음 식(1)과 같이 정의한다.

$$\sum_{n=0}^{\infty} \gamma^n r_{p+n} \qquad (1)$$

로보트가 상태를 감지하고 행동하는 1 샘플링 시간을 스텝(step)으로 정의하면 r_t는 로보트가 상태 s로부터 출발하여 행동책략 f를 따라 진행할 경우 어떤 스텝 p에서 받을 보답이라고 정의된다. 식(1)에서 γ 는 시간 축에 따라 감소하는 감쇠 상수이며 먼 미래의 보답이 행동책략에 얼마 만큼의 영향을 끼칠 것인가를 정하기 위해 사용된다. 대개는 1 이하의 값으로 정의된다. 만일 상태전이확률들과 각 상태전이에 대한 보답분포(reward distribution)를 미리 알 수 있다면 잘 알려진 dynamic programming[3]에 의해 최적의 행동책략(policy)를 구할 수 있을 것이다. 그러나 이러한 정보를 알 수 없으므로 Wakin은 Q-learning을 개발하게 되었다. $Q'(s, a)$는 어떤 상태 s에서 행동 a를 취하고 이 후에 최적의 행동책략 (optimal policy) f를 따르기 위한 응답값(return value) 혹은 행위값(action-value)이라 하고, 다음과 같이 정의 된다.

$$Q'(s,a) = r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a' \in A} Q'(s',a') \qquad (2)$$

Wakin은 초기에는 T와 r에 대해 아는 바가 없으므로, 최적의 Q값으로 점증적으로 접근해가기 위해 온라인(on-line)으로 Q값을 산정함으로써 식 (2)의 역할을 할 수 있도록 하고자 하였다. 이러한 Q값의 갱신은 다음 식 (3)와 같이 정의되었다.

$$Q(s,a) = \alpha Q(s,a) + (1-\alpha)(r(s,a) + \gamma \max_{a' \in A} Q(s',a')) \qquad (3)$$

여기서 보답 r은 상태 s에서 행위 a를 수행하는 것에 대한 실질적인 보답값(reward value)이고 s'은 다음상태를, α ($0<\alpha<1$)는 학습속도를 각각 나타낸다. 이러한 Q-learning은 다음과 같이 요약할 수 있다. 일단 Q값이 갱신되면 현재상태에서의 각 상태에 대한 모든 행위의 Q값을 저장하고 있는 Q-table를 교정하고, Q-table를 바탕으로 policy table을 식 (4)와 같이 교정해야 한다.

$$f(s) \leftarrow a \ such\ that \ Q(s,a) = \max_{b \in A} Q(s,b) \qquad (4)$$

위에서 정의한 Q value 갱신 규칙을 기본으로 한 Q-learning Algorithm은 다음과 같다.

Q-learning Algorithm

1. Initialization : Q table initialization with random value or prior information → policy table initialization
2. Receive s ← current state
3. Select an action a based on policy table but an random action with random action ratio ρ.
4. Excute action a, and receive reward r.
5. Update $Q(s,a)$ by equation (3).
6. Update the policy $f(s)$ (4).

3 Fuzzy Q-learning algorithm

3.1 Fuzzy Q-learning algorithm

Fuzzy Q-learning 은 앞 장에서 제시한 Q-learning 을 연속 공간상으로 확장한 학습 방법이다. Q-learning 에서처럼 공간상의 모든 상태에 대해 학습할 필요가 없고 몇 개의 대표적인 상태들에 대한 최적의 행위들을 학습하여 이의 fuzzy interpolation 으로 현재상태에서의 action 을 추론하는 방법이다. 앞으로의 설명을 위해 다음과 같은 표기를 사용하기로 한다.

$m_{c(n)i}$ ≡ 현재(다음)상태가 i 번째 주변상태로 소속될 소속도.
$a_{c(n)}$ ≡ 현재(다음)상태 최대의 Q 값을 갖는 action.
$a_{c(n)i}$ ≡ 현재(다음)상태의 i 번째 주변상태에서 최대의 Q 값을 갖는 action.
$s_{c(n)}$ ≡ 현재(다음)상태.
$s_{c(n)i}$ ≡ 현재(다음)상태에서 i 번째 주변상태.
$Q_{c(n)i}$ ≡ 현재(다음)상태의 i 번째 주변상태에서 최대의 Q 값.
$Q^a_{c(n)i}$ ≡ 현재(다음)상태의 i 번째 주변상태에서 현재 action a 가 갖는 Q 값.

일반적으로, Q-learning 에서는 최대의 Q 값을 갖는 행위가 하나인 경우는 문제가 없지만, 이러한 action 들이 여러 개인 경우는 이들 중 어떤 하나의 action 을 선택하여 학습하게 된다. 이와 같이, Fuzzy Q-learning 에서도 최대의 Q 값을 갖는 행위는 오직 하나라고 가정할 때 특정 상태에 대한 최적의 특정 행위로의 대응이 가능하다. 또한 상태 공간의 특성상 중간적인 상태에 대한 정의가 애매하므로 현재 상태를 주변상태들의 fuzzy interpolation 으로 정의 한다면, 주변에 존재하는 상태들로의 소속도에 의해 그림 2 처럼 현재상태가 정의된다.

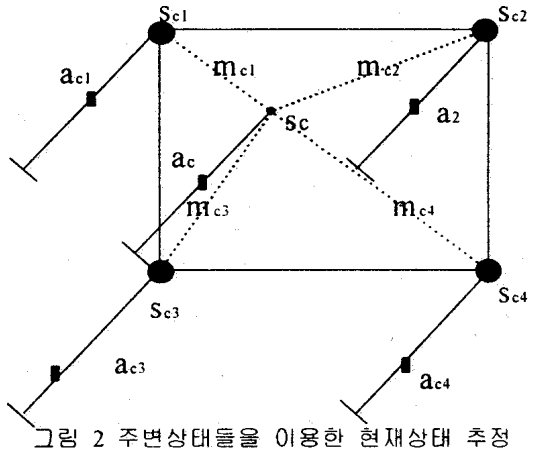


그림 2 주변상태들을 이용한 현재상태 추정

여기서, 소속도를 나타내는 membership 함수는 bell-shaped function 으로 식(5)와 같이 정의한다.

$$m_{ci} = e^{-\alpha \| s_{ci} - s_c \|} \qquad (5)$$

(여기서, α 는 scale 상수이고, s_{ci} 는 i 번째 주변상태벡터이며, s_c 는 현재상태벡터이다)

또한, 식 (5)을 이용하여 현재상태와 현재 취해야 할 행위를 각각 식(6), (7)로 정의 할 수 있다.

$$s_c = \sum_{i=1}^{N} m_{ci} s_{ci} \ (\text{여기서, N 은 주변상태의 수}) \qquad (6)$$

$$a_c = \sum_{i=1}^{N} m_{ci} a_{ci} \quad (7)$$

현재(다음)상태에서의 Q-value, $Q_c(Q_n)$를 현재(다음) 상태 및 현재(다음 step 에)취해야 할 행위를 추정하는 것과 같이 fuzzy interpolation 으로 정의한다면, 각각 다음 식 (8), (9)와 같이 나타낼 수 있다.

$$Q_c = \sum_{i=1}^{N} m_{ci} Q_{ci} \quad (8)$$

$$Q_n = \sum_{i=1}^{N} m_{ni} Q_{ni} \quad (9)$$

모든 상태에서 모든 행위들의 행위값을 나타내는 것이 Q-table 이고, 이를 현재상태 에서 Q-value 최고점을 기준으로 거리에 비례적으로 단순 감소하는 특성이 있는 함수로 modeling 하면, 특정 상태에서 모든 행위들에 대한 Q-table 를 그림 3 과 같이 cone-shaped function 으로 나타낼 수 있다. 그림 3 은 주변상태에서 최적의 행위와 이를 이용하여 현재상태에서 식(7)에 의해 추론된 현재행위를 나타내고 있다.

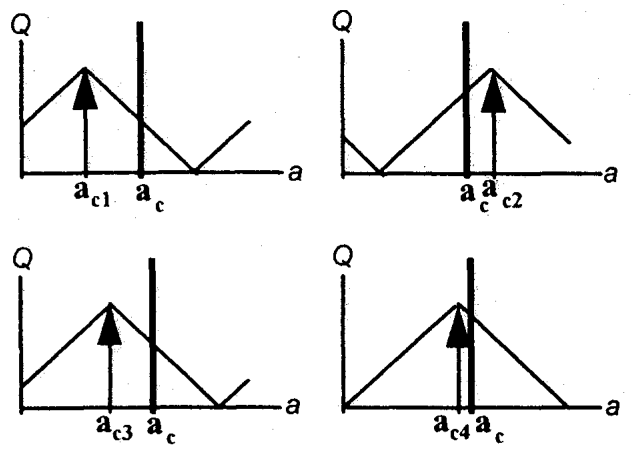


그림 3 현재상태의 적절한 행위 생성

현재행위를 수행한 후, 환경은 현재상태는 다음상태로 바뀌게 되고, 로보트는 외부로부터 보답(reward)을 수신하게 된다. 수신된 reward 를 이용한 Q-value 의 갱신은 다음 식 (10)과 같이 수행된다.

$$Q_c(t+1) = (1-\alpha) Q_c(t) + \alpha (r(t) + \gamma Q_n(t)) \quad (10)$$

(여기서, t 는 t 번째 iteration 을 나타낸다.)

식 (8)과 식(9)을 식(10)에 대입하여 다음 iteration 에서의 Q-value 를 산정할 수 있다. 여기서 본 알고리즘의 핵심인 주변상태의 최적 행위 수정을 통한 현재상태의 행위 추측을 위해, 역으로 식(10)에 의해 갱신된 현재상태에서 Q-value 갱신 양을 기본으로 주변 상태의 Q-value 를 얼마나 갱신 시켜야 하는지를 알아낸다. 따라서 식 (11)에 의해 i 번째 주변상태에 대한 현재행위의 Q-value 갱신 정도를 소속도를 고려하여 추정한 후, 식 (12)을 이용하여 i 번째 주변상태에서의 현재행위 a 에 대한 Q-value 를 갱신한다.

$$dQ_{ci}^{a}(t+1) = m_{ci} (Q_{ci}^{a}(t+1) - Q_{ci}^{a}(t)) \quad (11)$$

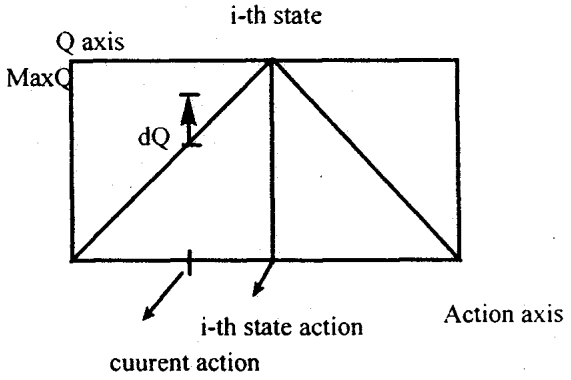
$$Q_{ci}^{a}(t+1) = Q_{ci}^{a}(t) + dQ_{ci}^{a}(t+1) \quad (12)$$

그림 4, 5 는 식 (11), (12)을 근거로 하여, 현재행위에 대해 i 번째 주변상태의 Q-table 내의 현재행위에 대한 Q 값 갱신 방법을 나타낸다. 먼저, 그림 5 는 갱신된 Q 값이 현재의 Maximum Q 값 보다 작은 경우, 갱신된 Q 값에 대해 적응하기 위해 cone 모양의 Q-table model 이 왼쪽으로 움직이는 width modification 을 나타내며, 식 (13)으로 표현된다. 이를 통해 주변상태에서의 최적의 행위가 수정 된다.또한, 그림 6 은 갱신된 Q 값이 현재의 Maximum Q 값 보다 큰 경우, 갱신된 Q 값에 대해 적응하기 위해 cone 모양의 Q-table model 이 현재행위쪽으로 움직이는 height modification 을 나타내며, 식 (14)로 표현된다. 이 경우, 주변상태에서의 최적 행위는 현재행위가 된다.

$$a_{ci}(t+1) = a_{ci}(t) + \mathrm{sgn}(\bullet) 2 \frac{a_{max}}{Q_{max}} dQ \quad (13)$$

$$(\text{여기서, } \mathrm{sgn}(\bullet) = \mathrm{sgn}(\frac{a_{ci}(t+1) - a_{ci}(t)}{dQ}))$$

$$a_{ci}(t+1) = a_c \quad (14)$$



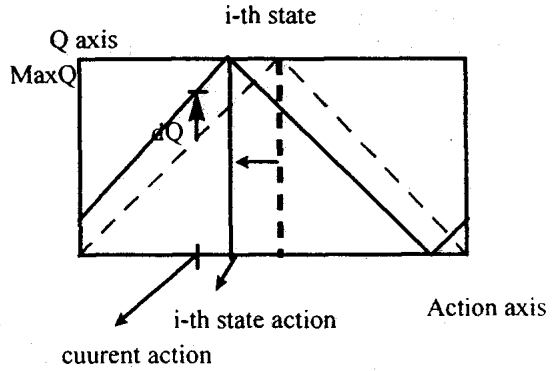
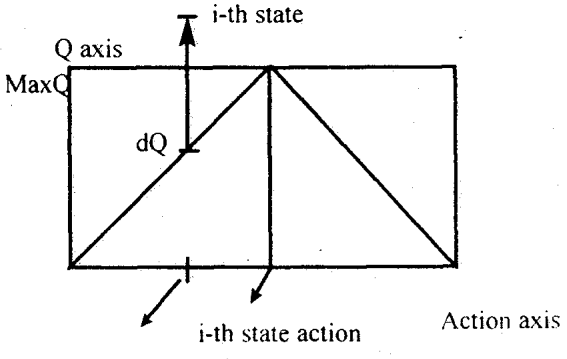


그림 4 Q-value update (width modification)



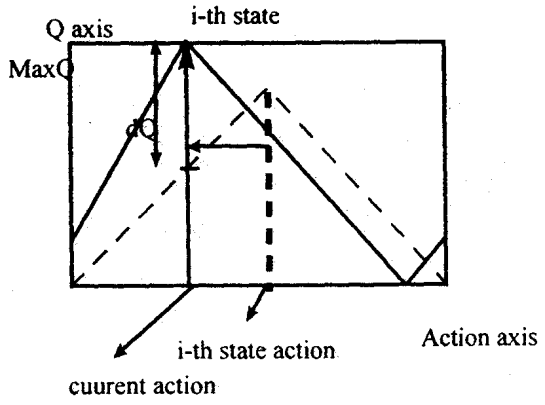


그림 5 Q-value update (height modification)

이와 같이 갱신된 주변 상태들의 Q 값에 근거하여 다음 행위을 취하게 된다. Fuzzy Q-learning algorithm 은 다음과 같이 정리할 수 있다.

Fuzzy Q-learning Algorithm

1. 상태 s 가 주변상태에 어느정도 속하는지를 알 수 있도록 한다. (Membership (mi)를 계산한다.)
2. 상태 s 에서 실행해야 할 행위는 식 (19)에 의해 구한다. (때로는 Random action 수행)
3. 결정된 행위을 수행하고, Reward 를 받는다.
4. 현재상태에서 Q 값은 식(10)를 사용하여 구한다.
5. 주변상태들에서의 Q 값은 식(11-12)를 사용하여 구한다.
6. 현재상태에서의 최적의 행위을 식(13)(14)을 사용하여 갱신한다.

Fuzzy Q-learning 은 Q-learning 에 비해 다음과 같이 특징을 갖는다.

1. 연속상태공간을 대상으로 한다.
2. 모든 상태들을 정의할 필요가 없다. (일정 수의 상태만을 정의하고 미정의 상태들은 주변상태들로의 소속 정도를 이용하여 추론한다.)
3. 연속된 행위 모델을 사용하여 정의된 상태에서는 최적의 행위만을 기억한다.
4. 매번 행위을 선택할 때마다 모든 행위에 대하여 Maximum Q Value 를 계산하여 그 Q Value 에 해당하는 행위을 실행

4 모의 실험

4.1 Q 와 FQ learning algorithm 의 비교 모의실험

Q-learning 과 Fuzzy Q-learning 의 성능을 비교해보기 위해 자유 공간상에서 현재 위치에서 원하는 위치까지 도달하기 위한 policy 를 구하는 simulation 을 실시하였다. 먼저 simulation 의 환경은 다음 표 1 과 같다.

표 1 Simulation 환경 설정

Row state axis	0 - 350 state space
Column state axis	0 - 350 state space
Action	8 방향 vector (Q) 연속 vector (FQ)
Sampling Time	0.032 sec

본 simulation 에서 사용된 reward r 은 "Move to goal"을 기준으로 다음 식 (15)와 같이 정의한다

$$r = \|x - g\| - \|y - g\| \qquad (15)$$

(여기서, 현재 position = x, 다음 position = y, 최종 position = g)

초기 위치를 (50,50)으로 하고 최종 위치(320,320)라 할 때, Q-learning 의 경우 position 상태 공간을 각 축마다 35 개의 resolution 으로 나누어야 목표 지점에 도달할 수 있다. 그림 6, 7 은 아무런 사전 정보 없이 이동하는 초기 iteration 에서는 여러 방향으로 탐색하는 과정을 보여준다. 그러나 그림 8, 9 에서와 같이 Q 의 경우 약 400 번의 iteration 을 수행한 뒤에 원하는 상태 근처로 수렴하는 반면에, FQ 의 경우 100 번의 iteration 후에 수렴하는 것을 볼 수 있다. 그러나 Q, FQ 양쪽 모두, 많은 iteration 을 수행한 후에도 수렴하지 않는 경우가 있는데 이는 reward 와 parameter (α, γ)설정이 잘 못된 경우이다. 예로써, 상태 s 에서 현재행위 $a1$ 에 대한 reward 를 $r1$ 이라고 하고, optimal 행위, $a2$ 의 reward 를 $r2$ 라고 하자. Optimal 행위 $a2$ 가 policy 에 등록되기 위해서는 Q value 가 가장 커야 하므로 식 (12)에서와 같이 갱신을 반복하는 과정에서 현재 Q-value 의 차이를 γ 혹은 α 을 사용하여 극복하여야 한다. 즉, γ (α)를 제외한 나머지 parameter 를 상수라고 하면 두 행위에 대한 Q-value update 속도는 γ (α)에 비례하여 결정된다. 따라서 적절한 γ (α)의 선택이 선행되어야 한다. 또한 Q 와 FQ 의 iteration 별 step 수는 약 40 번 내외로 수렴됨을 알 수 있었다. Fuzzy Q-learning simulation 결과 우수한 수렴 속도와 Q-learning 보다 부드러운 action set 을 학습함을 알 수 있다. 이와 더불어 Fuzzy Q-learning 의 또 다른 특징인 적은 상태를 정의하고 비슷한 성능발휘을 그림 10 에서 알 수 있다. 즉, 그림 10 에서는 Fuzzy Q-learning 의 경우 적은 상태(각 축당 17 상태)을 정의하고도 원하는 상태로 수렴됨을 보여주고 있다. 이 경우에서도 iteration 당 수렴 step 수는 약 40 정도에서 수렴함을 알 수 있었다.

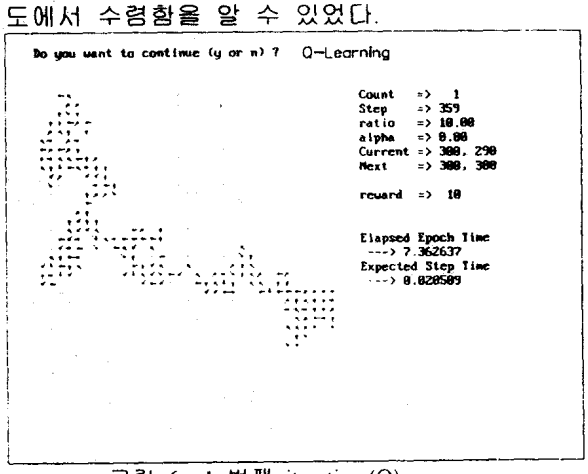


그림 6 1 번째 iteration (Q)

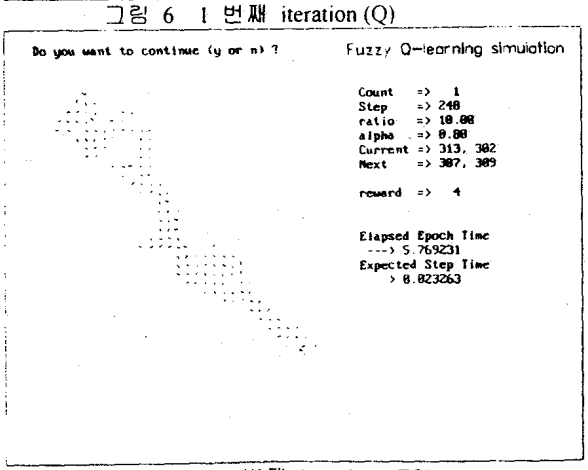


그림 7 1 번째 iteration (FQ)

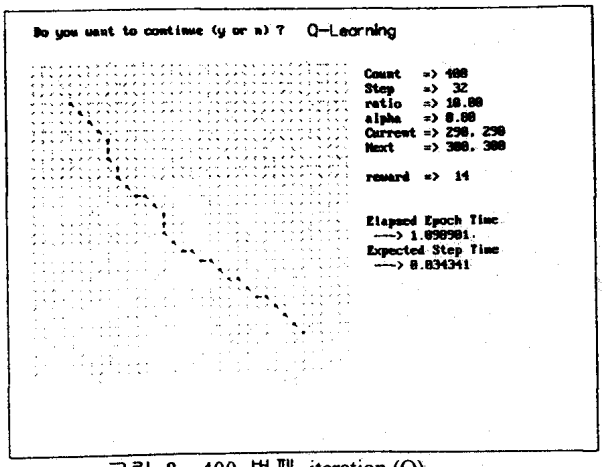


그림 8 400 번째 iteration (Q)

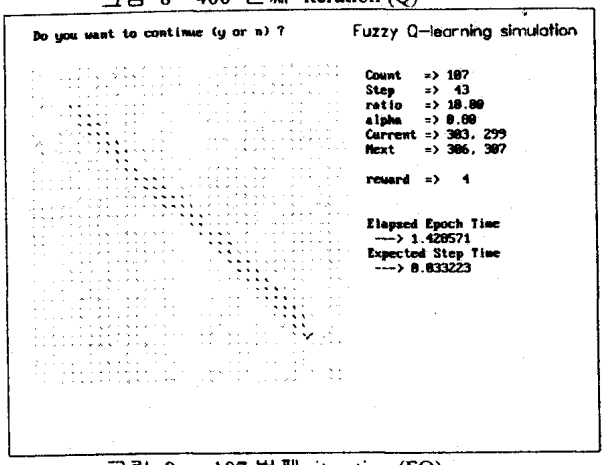


그림 9 107 번째 iteration (FQ)

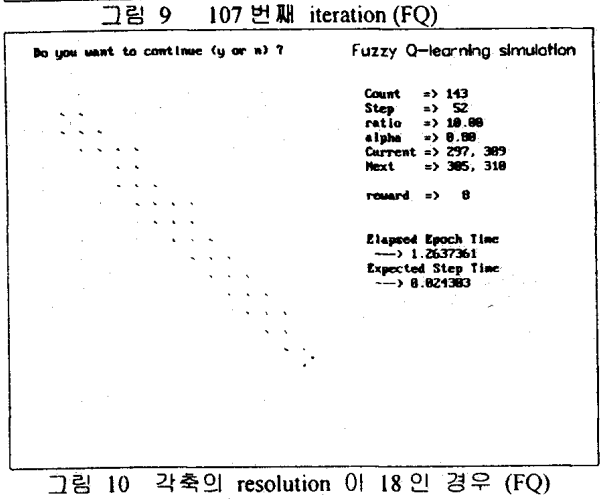


그림 10 각축의 resolution 이 18 인 경우 (FQ)

4.2 Box Canyon Example

Fuzzy Q-learning 을 적용하기 위한 일반적인 응용 분야는 상당히 다양할 수 있지만, 일반적으로 많이 알려져 있는 box canyon problem 을 simulation 대상으로 하여 개발된 algorithm 의 유용성을 보이고자 하였다. 먼저, 사용된 Reward 는 다음과 같다.

Reward = 식 (15) - Move step size $(1-e^{-2visit})$ (16)

(여기서 visit 은 동일 상태를 탐색한 횟수를 나타냄.)

초기에는 "Go to goal" reward, 식(15)에 의해 목적지에 대한 reward 가 큰 작용을 하여, box canyon 의 골짜기에 빠져 이리 저리 탐색하게 되지만, 일단 상태들이 visit 된 수가 높아지면 reward 의 두번째 항이 증가하고, 따라서, 전체 reward 는 상대적으로 낮아지게 된다. 궁극적으로는 그림 11 에서처럼 원하던 목적지에 도달하게 된다. 그림 12 은 iteration 이 8 회 지난 후에, 각 출발 상태로부터 목적상태쪽으로 학습이 수렴됨을 나타낸다. 여기서 Move step size 는 일정한 것으로 간주 하였지만 실질적인 실험에서는 Fuzzy Q-learning 은 방향벡터를 생성하는 것으로 속도 벡터의 크기는 현재 위치로부터 목표물까지 혹은 장애물까지의 거리를 특정할 수 있는 초음파 sensor data 혹은 vision sensor data 에 따라 조절되어야 한다.

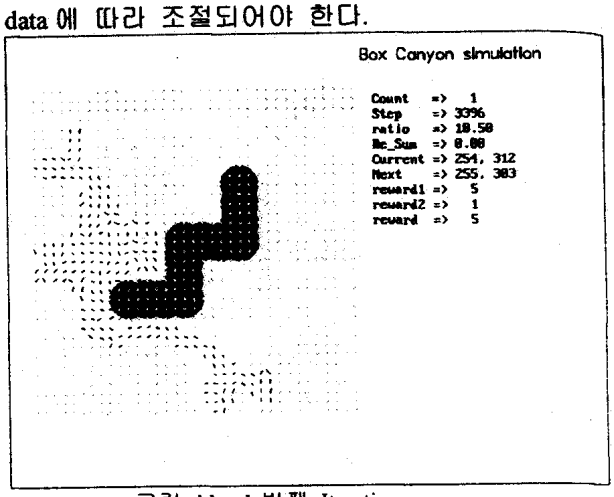


그림 11 1 번째 Iteration

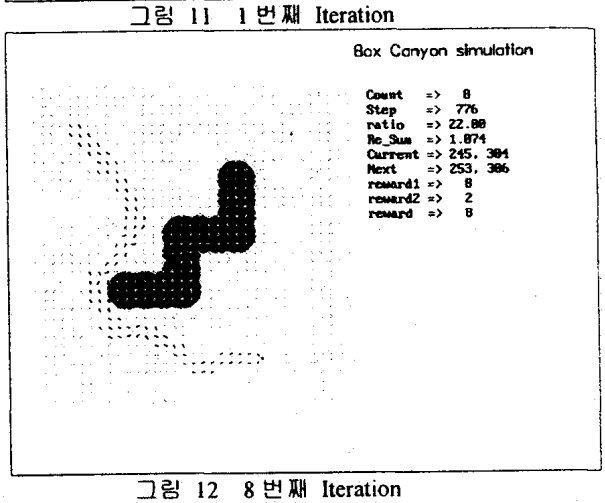


그림 12 8 번째 Iteration

5 결론

본 논문에서는 연속 상태 공간에서 각 상태에 적절한 연속된 행동 양식을 학습하기 위해서 Fuzzy Q-learning 알고리즘을 제안 하였다. 또한 이의 효율성을 증명하기 위해 기존의 Q-learning 알고리즘과의 simulation 을 수행하였다. 본 연구는 앞으로 Real system 개발등 많은 환경에 대한 정보가 부족한 상태에서 연속 공간에 대한 연속된 행위를 수행하여야 하는 응용 분야에 효과적으로 적용될 수 있을것으로 판단 된다. 향후에는 좀 더 적절한 parameter 조합을 이루기 위해서는 learning 상태에 알맞는 parameter 를 자동으로 결정하는 알고리즘에 대한 연구가 진행되어야 할 것이다.

References

[1] H. Berenji, "Reinforcement learning and recruitment mechanism for adaptive distributed control", IR/IRIDIA/92-4, Universite Libre de Bruxelles, 1992.

[2] C. Watkins, P. Dayan, "Q-learning, Technical Note", Maching Learning, Vol. 8, pp.279-292, 1992.

[3] R. Bellman. Dynamic Programming. Princeton University Press, Priceton, NJ. 1957.

[4] L. J. Lin, "Programming robots using reinforcement learning and teaching," *In Proc. of the Ninth National Conferernce on Artificial Intelligence*, 1991.