함수근사와 규칙추출을 위한 클러스터링을 이용한 강화학습

(Reinforcement Learning with Clustering for Function Approximation and Rule Extraction)

이 영 아 † 홍 석 미 †† 정 태 충 †††
(YoungAh Lee) (SeokMi Hong) (TaeChoong Chung)

요 약 강화학습의 대표적인 알고리즘인 Q-Learning은 상태공간의 모든 상태-행동 쌍(state-action pairs)의 평가값이 수렴할 때까지 반복해서 경험하여 최적의 전략(policy)을 얻는다. 상태공간을 구성하는 요소(feature)들이 많거나 요소의 데이타 형태가 연속형(continuous)인 경우, 상태공간은 지수적으로 증가하게 되어, 모든 상태들을 반복해서 경험해야 하고 모든 상태-행동 쌍의 Q값을 저장하는 것은 시간과 메모리에 있어서 어려운 문제이다. 본 논문에서는 온라인으로 학습을 진행하면서 비슷한 상황의 상태들을 클러스터링(clustering)하고 새로운 경험에 적용해서 클러스터(cluster)의 수정(update)을 반복하여, 분류된 최적의 전략(policy)을 얻는 새로운 함수근사(function approximation)방법인 Q-Map을 소개한다. 클러스터링으로 인해 정교한 제어가 필요한 상태(state)는 규칙(rule)으로 추출하여 보완하였다. 미로환경과 마운틴 카 문제를 제안한 Q-Map으로 실험한 결과 분류된 지식을 얻을 수 있었으며 가시화된(explicit) 지식의 형태인 규칙(rule)으로도 쉽게 변환할 수 있었다.

키워드 : 강화학습, Q-학습, 클러스터링, 근사함수, 규칙 추출, 자기조직형상화 지도

Abstract Q-Learning, a representative algorithm of reinforcement learning, experiences repeatedly until estimation values about all state-action pairs of state space converge and achieve optimal policies. When the state space is high dimensional or continuous, complex reinforcement learning tasks involve very large state space and suffer from storing all individual state values in a single table. We introduce Q-Map that is new function approximation method to get classified policies. As an agent learns on-line, Q-Map groups states of similar situations and adapts to new experiences repeatedly. State-action pairs necessary for fine control are treated in the form of rule. As a result of experiment in maze environment and mountain car problem, we can achieve classified knowledge and extract easily rules from Q-Map

Key words : reinforcement learning, Q-Learning, clustering, function approximation, rule extraction, self-organizing features map

1. 서 론

강화학습(reinforcement learning)의 대표적인 알고리즘인 Q-학습(Q-learning)은 상태공간(state space)의 모든 상태-행동 쌍(state-action pairs)을 반복적으로 경험하면서, 환경(environment)이 주는 보상값(reward)과 평가값인 Q값(Q-value)을 기반으로 상태-행동 쌍에 대한 전략(policy)을 학습한다. 강화학습의 대표적인 방법인 Q-학습의 특징을 몇 가지 정리하면 다음과 같다. 첫째, 만일 임의의 상태에서 어떠한 행동을 취했을 때 목표 상태(goal state)에 도달한다면, 큰 강화값을 받게 되어 해당 상태-행동 쌍의 Q값이 크게 갱신된다. 1-step Q-학습의 Q값 갱신식(식 (1))은 현재(t) 경험한 상태-행동 쌍의 Q값을 갱신하기 위하여, 다음 상태(t+1)에서 가능한 행동들이 갖는 Q값 중에서 가장 큰 Q값을 이용하므로, 일단 목표 상태에 도달했었다면 목표까지의 경로 상에 있는 상태-행동 쌍의 Q값은 영향을 받게된다.

† 학생회원 : 경희대학교 컴퓨터공학과
leeyaa@iislab.kyunghee.ac.kr
†† 학생회원 : 경희대학교 컴퓨터공학과
smhong@yahoo.co.kr
††† 종신회원 : 경희대학교 컴퓨터공학과 교수
tcchung@khu.ac.kr
논문접수 : 2002년 12월 28일
심사완료 : 2003년 7월 30일

$$Q(input, a) = Q(input, a) + \alpha[r + \gamma \max_{a'} Q(input', a') - Q(input, a)]$$

(α : learning rate, γ : discount factor) (1)

둘째, 목표 상태에 도달했을 때와 마찬가지로, 임의의 상태에서 취한 행동에 의해 치명적인 다음 상태(예, 격자공간에서 장애물이 있는 상태 또는 격자공간의 밖)로 전이하게 된다면, 해당 상태-행동 쌍은 매우 작은 강화값을 받게 되고 Q값 또한 낮게 갱신된다. 그리고 치명적인 상태로 가는 경로 상에 있는 상태-행동 쌍의 Q값은 영향을 받게 된다. 셋째, 문제의 요소(feature)들이 연속값(continuous value)을 갖거나 고차원(high-dimensional)인 경우, 상태공간이 거대해지기 때문에, 모든 상태-행동 쌍에 대해 Q값이 수렴할 때까지 경험을 반복한다는 것은 불가능하다. 이 문제를 해결하기 위해서, 유사한 상태-행동들을 일반화하는 신경망(neural net), CMAC(cerebella model articulation controller)와 같은 함수근사(function approximator)를 이용한다[1-3]. 신경망은 분류(classification)와 함수근사가 필요한 응용문제를 해결하기 위해서 성공적으로 이용되어 왔다. 그러나 훈련된 신경망은 최적의 전략을 구할 수는 있지만, 블랙박스이므로 문제가 어떻게 해결되는가를 설명하는 가시적인 규칙(rule)을 쉽게 추출할 수 없다. 규칙추출(rule extraction)은 몇 가지 장점을 가지고 있다. 첫째, 학습하는 과정에서 추출된 규칙은 학습 속도를 높일 수 있고, 둘째, 결정적인 상태에서 치명적인 행동은 피하고 다른 행동을 탐험(exploration)할 수 있도록 인도하며, 셋째, 전문가(human experts)가 학습된 지식을 이해하는데 도움을 준다[3,5-7].

본 논문에서는 Q-학습을 진행하면서 경험하게 되는 상태-행동 쌍들을 유사정도(similarity)에 따라 클러스터링하고, 계속해서 클러스터(cluster)의 중심값(center)을 반복되는 경험에 적응시켜 나가는 근사 방법(approximation method)인 Q-Map을 제안한다. 승자(winner)가 되는 클러스터의 중심값은 입력 상태와 중심값의 오차와 학습율(learning rate)에 의해 갱신된다. Q-Map은 초기화하기 위한 사전 지식이 필요 없고, 유사한 상태들의 중심값만을 저장하므로 크기가 작고, 학습과 동시에 상태공간을 지역적으로(locally) 분석한 결과를 얻게 된다. Q-Map의 각 클러스터는 가시적인(explicit) 지식인 규칙(rule)으로 직접(directly) 추출할 수 있다. 또한 유클리드 거리를 이용하여 유사정도를 구하고 오직 승자의 중심값만을 갱신하므로, Q-Map을 구성하기 위해 필요한 계산량이 적다. 본 논문에서는 장애물이 있는 격자공간에서 목표까지 가기 위한 전략을 학습하는 문제와 요소(feature)들이 연속값을 갖는 마운틴 카(mountain car)문제에서 최적에 근사하는 전략을 구하기 위하여 Q-Map을 실험하여 보았다. 마운틴 카 문제인 경우, 요소의 연속값을 일정한 간격으로 세밀하게 나누어(quantization) 기본적인 Q-학습을 실행한 결과와 각 클러스터를 임의의 값으로 초기화한 후 계속해서 중심값을 적응해 나가는 Q-Map의 결과를 비교해 보았다.

2. 관련 연구

강화학습에서 상태공간이 매우 큰 경우, 학습을 가속시키고 결과를 향상시키기 위해서 모듈러 강화학습(modular reinforcement learning)과 명시적인 지식의 표현이 필요하다. Q-학습에서는 모든 상태-행동의 값을 저장하기 위하여 룩업 테이블(lookup table)을 이용하는데, 문제(task)에 따라서 테이블의 크기가 매우 거대해질 수 있고, 학습한 모든 내용을 담고 있지만 각 상태-행동에 대한 단위 정보이고 분석된 정보라고는 볼 수 없다. 거대한 상태공간을 탐험해야 하는 경우, 대부분의 방법들은 상태공간을 작은 영역들(regions)로 나누고 각 영역의 전략은 함수근사 방법(function approximation method)으로 일반화 시킨다. 함수근사방법으로는 일반적으로 역전파 신경망(back-propagation neural net)을 이용하는데, 큰 상태공간을 작은 모듈들로 나누고, 각 모듈의 Q-학습을 역전파 신경망으로 구현한다.[4,5] 모듈을 나누는 과정은 선행처리 되거나 강화학습을 진행하면서 자동으로 나누게 된다. 각 모듈을 처리하면 복잡도(complexity)는 낮아지겠지만, 역전파 신경망은 블랙박스로서 입력에 따른 결과만 나오므로 가시화된 지식을 표현하기 어렵다.

강화학습을 적용하는 많은 문제들(tasks)은 데이타의 분포(data distribution)와 분류정보가 사전에 주어지지 않으므로, 함수 근사 방법으로 비교사 학습(unsupervised learning)방법인 논 파라메트릭 클러스터링(non-parametric clustering)이 적합하다. 논 파라메트릭 클러스터링은 파라미터를 적응하기 위한 학습이 필요 없고 클러스터들을 구성하기 위한 제약조건만을 따르면 된다. 그러나 기존의 클러스터링 알고리즘을 그대로 강화학습에 이용할 수는 없다. LVQ(Learning Vector Quantization)는 데이타와 함께 주어지는 분류정보(classifier)를 이용하여 클러스터의 중심을 적응시키므로 상태-행동 쌍에 대한 보상값(reward)만이 피드백 되는 강화학습에 그대로 적용할 수 없다. 또한 자기조직화 형상 지도(Self-Organization Features Map, SOM)는 특정한 입력에 대해 가장 잘 반응하는 노드와 이웃의 노드를 학습시키는데, Q-학습은 하나의 클러스터 내에서도 각 행동별로 학습이 이루어져야 하고, 행동에 따라 이웃이 변하므로 SOM의 이웃개념을 이용하기 어렵다[4].

논 파라메트릭 클러스터링은 유사한 상태들의 집합체

(state aggregation)를 만드는 방법과 결정 한계선(decision boundary)을 찾아서 상태공간을 분할(resolution)하는 방법[1]으로 나눌 수 있다. 분할방법으로는 Stuart I. Reynolds의 Decision Boundary Partitioning[1]을 예로 들 수 있다. 큰 분할(coarse)부터 시작해서 하나의 영역에서 최적의 행동이 다르면 반복해서 분할하는 방법이다. 상태공간의 일부 영역에서 전략이 세분화되는 문제라면 보다 효과적일 수 있다. Sun과 Peterson(1999)이 개발한 CLARION 프레임웍[5,7]은 분할(resolution)과 Q-학습, 그리고 규칙 학습을 두 단계로 처리한다.(Sun, 1997) 하위 단계에서는 이들이 제안한 영역분할방법(region-splitting method)을 이용하여 온라인으로 복잡한 강화학습 문제를 작은 모듈들로 나누고, 각 영역을 역전파 신경망으로 구현한다. 상위 단계에서는 하위단계에서 s상태에서 선택한 행동 a가 매우 성공적이었다면(즉, $r+\gamma \max_b Q(y,b)-Q(x,a) > threshold$), 규칙 s→a를 추출 한다. 규칙의 조건부에서 각 요소 x_i는 하나의 값을 갖거나($x_i=v_i$) 또는 $x_i \in \{v_{i1}, v_{i2}, \ldots\}$가 된다. 식 ($r+\gamma \max_b Q(y,b)-Q(x,a) > threshold$)에서 선택한 행동이 성공적이었는지를 평가하기 위해서 임계값(threshold)은 사용자의 사전지식에 의해 정해져야 한다. 규칙들은 학습을 진행하면서 적용되고 검증된다. 만일 적용된 규칙이 효과적이었다면 규칙의 조건부(condition part)는 확장되는데, 조건부의 한 요소(feature)에 값을 추가하여 여러 상태에서 규칙을 적용할 수 있도록 한다. 반대로 적용 결과가 부정적이라면 규칙을 축소하기 위해서 한 요소의 값을 제거하여 적용되는 경우를 줄인다.

본 논문에서 제안하는 Q-Map은 각 클러스터의 중심값과 가능한 각 행동의 Q값을 기억하는 구조이다. 상태공간의 분할을 선행처리 하는 경우에는 도메인을 잘 아는 유저의 사전지식이 필요하고 분할이 고정 된다. 반면 Q-Map은 온라인으로 Q-학습을 진행하면서 각 클러스터의 중심값을 새로운 경험에 적응하므로 사전지식이 필요 없고, 분할은 조금씩 변하게 된다. Q-Map에서 각 클러스터의 중심값은 규칙의 조건부에 해당하는데, 유사한 상태들을 대표한다. 최종 학습된 결과를 담고 있는 Q-Map은 상태공간을 지역적으로 분석한 결과이고, 별도의 복잡한 규칙 추출과정이 필요 없다.

3. 경험 지식의 클러스터링과 규칙 추출

강화학습과 클러스터링은 사전 지식이 필요 없고, 예제가 아닌 경험과 관찰을 통해서 학습한다. 클러스터링을 이용하여 유사한 상태들의 전략을 학습하는 경우, 클러스터가 제안하는 행동과는 상관없이 결정적인 행동을 해야 하는 상태들을 구분할 수 있어야 한다. 본 논문에서는 특별히 구분되어야 하는 상태-행동 쌍을 별도의 규칙으로 관리한다. 일단 규칙이 추출되면, 행동을 선택할 때 이용되어 결정적인 상태에서 필요 없는 탐험(exploration)을 피할 수 있다. 규칙은 C->A(C : 조건부, A: 행동부)의 형태인데 A는 긍정적인 행동과 부정적인 행동으로 나눌 수 있다. 목표상태로 가는 행동인 경우, 결정적이므로 다른 행동들을 학습할 필요가 없다. 치명적인 상태로 전이하는 행동인 경우, 부정적으로 보고 나머지 행동에 대해서 Q값을 관리해서, 치명적인 행동은 피하고 가능한 행동들 중에서 최적의 행동을 학습해야 한다.

3.1 제안 모델의 구성

최적의 전략에 근사하는 방법인 Q-Map을 이용하여 Q-학습을 하는 제안 모델은 세 개의 모듈로 구성된다 (그림 1). 첫째, 상태공간을 탐험하면서 겪는 경험을 통

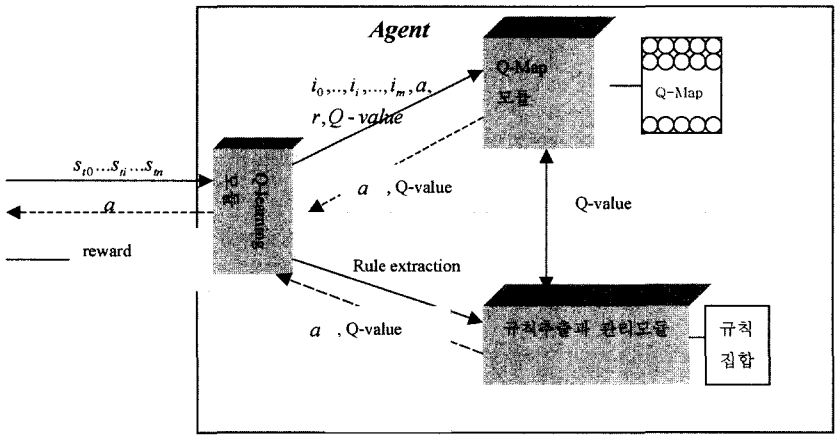


그림 1 제안 모델의 구성

해 학습하는 Q-학습 모듈이 필요하다. 둘째, 경험한 상태-행동 쌍의 Q값을 저장하고 클러스터를 관리하는 모듈이다. 모든 상태-행동의 쌍에 대한 Q값을 저장하는 대신, 각 클러스터의 중심에 해당하는 상태의 Q값만을 저장한다. 각 클러스터는 패턴들을 인지하기 위해 중심값을 기억하고 있는 뇌 구조라고 볼 수 있다. 각 클러스터의 중심은 코넨(Kohonen)의 자기조직화 형상 지도(Self-Organization Features Map, SOM)에서 각 뉴런과 입력 벡터사이의 가중치를 갱신하는 방법과 유사하게 관리된다. 세 번째 모듈은 규칙을 추출하고 관리한다.

3.2 Q-Map과 규칙

Q-Map은 그림 2와 같이 2차원으로, 행의 개수는 사용자가 지정한 클러스터의 개수이고, 열의 개수는 상태공간에서 가능한 행동의 수이다. 그리고 강화학습을 하는 동안 여러 에피소드들(임의의 상태에서 출발하여 목표 상태에 도달할 때까지의 경로)을 경험해야 하는데, 목표상태(goal state)는 한 에피소드의 끝으로서 다음 상태로 전이되지 않으므로 클러스터링된 다른 상태와 같이 다룰 수 없다. 그러므로 목표상태를 중심으로 갖는 클러스터를 추가하여, Q-Map을 구성하는 노드의 수는 (클러스터의 수 * 가능한 행동의 수) + 1이 된다.

Q-Map 관리 모듈은, Q값과 최적의 전략을 참조하고 Q값을 갱신하기 위하여, 유클리드 거리(식 2)를 이용해서 승자 클러스터를 찾는다.

$$d_c=\sqrt{\sum_{i=0}^{n}(weight_{ci}-input_i)^2} \qquad (2)$$

d_c: n차원 상태공간에서 입력 벡터와 중심 벡터사이의 유클리드 거리

$input_i$: 입력 벡터의 feature i의 값,

$weight_{ci}$:클러스터 c에서 중심 벡터의 feature i의 값

승자 클러스터는 d_c가 최소가 되는 클러스터이다. 승자 클러스터의 중심값은 식 (3)과 식 (4)와 같이 입력 상태와 승자 클러스터의 중심값의 오차와 학습률(learning rate)을 이용하여 갱신된다.

$$weight_i = weight_i + \alpha \times (input_i - weight_i) \qquad (3)$$

$\alpha = 0.5 \times 0.9^{\frac{t}{1000}}$ α : learning rate, t: 현재 시점 (4)

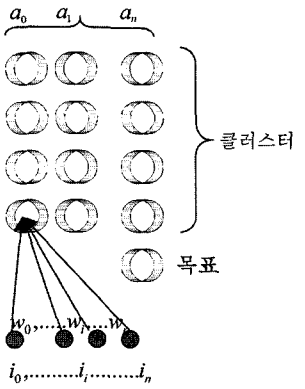
학습률(learning rate) α는 0.5로 초기화하고, 학습하는 동안 식 (4)에 따라 조금씩 감소된다. $t(1 \le t < \infty)$는 현재 시점으로서 현재까지 Q-Map에 적용된 입력 상태의 수를 의미한다.

임의의 상태 (s_t)에서 선택한 행동이 목표상태로 전이하는 경우와 치명적인 상태로 전이해서 페널티를 받는 경우를 규칙으로 추출한다. 규칙은 그림 3의 형식을 갖는데, C는 상태를 구성하는 요소(feature)들의 값으로 구성이 되고, A는 가능한 행동들의 집합이다. A의 각 행동들은 Q값을 가지고 있는데, 학습이 진행되면서 계속 갱신된다.

If C then A
C : Condition part, (x = val1 AND y = val2...)
A : Action part, {(a_1 with Q), (a_2 with Q), (a_3 with Q), (a_4 with Q)}

그림 3 규칙의 형태

그림 1은 제안모델의 구성과 동작을 보여준다. Q-학습모듈로 상태정보인 입력벡터($s_{t0}, .. s_{ti}, .. s_{tn}$)가 들어오면, 전략에 따른 행동(a)을 선택하기 위하여 Q-Map과 규칙 집합(rule set)을 참조하게 된다. 또는 ε-greedy 정책을 따라서 전략과 상관없이 랜덤하게 행동



Input vector: i_0,....i_i,....i_n
Center (weight vector): w_0,..... w_i.... w_n
n : number of features of state

a cluster:

w_0	...	w_i	w_n	action	reward	Qvalue

그림 2 Q-Map

을 선택한다. 이 과정에서 목표에 도달하거나 치명적인 다음 상태로 전이한다면, 해당 상태-행동쌍을 규칙으로 추출하게 된다. 전략을 따른다면, 먼저 규칙집합에서 현재(t)의 상태에서 목표로 갈 수 있는 행동이 가능한지, 또는 치명적인 행동이 있는가를 검색한 후, 해당하는 규칙이 있다면 실행한다. 규칙집합에 실행할 규칙이 없다면, Q-Map에서 현재 상태와 유클리드 거리상 가장 가까운 승자 클러스터를 결정하고, 승자 클러스터가 제안하는 행동(a)을 취한다. 취한 행동에 대한 보상값(reward)과 식 (1)의 Q값 갱신식을 이용하여 승자 클러스터의 Q값을 갱신하는 과정을 반복한다.

제안 모델의 마지막 단계는 Q-Map의 각 클러스터를 규칙으로 변환하는 것이다. 클러스터의 중심과 규칙집합(rule set)에서 조건부(C)가 같은 규칙이 존재한다면, 규칙을 따르고 해당 클러스터들은 제거한다.

제안 모델의 알고리즘은 정리하면 다음과 같다.

4 실험과 분석

4.1 장애물이 있는 격자공간

본 연구는 우선 상태공간의 요소가 이산값(discrete value)을 갖는 격자공간(grid world)에서 Q-Map을 실험하였다. 문제의 목적은 격자공간에는 장애물들이 놓여 있고, 임의의 셀에서 출발하여 장애물을 피하면서 목표 셀까지 가는 최단 경로를 구하는 것이다. 상태(state)는 x좌표와 y좌표로 구성된다. 각 상태(s_t)에서 수행할 수 있는 행동의 집합은 $A(s_t)=\{E, W, S, N,\}$이다. Q-Map의 클러스터의 중심값은 랜덤하게 또는 전처리 과정으로 얻은 값들로 초기화했다. Q-Map의 적응력을 보기 위해 여러 형태의 장애물이 있는 격자공간(그림 5, 그림 6)에서 실험하였고, 각 에피소드의 시작 상태는 임의로(randomly) 선택하였다. 그림 6에서 x=7, y=3인 상태는 S(south) 행동만 가능한데, 행동 N, E, W를 취해보고

```
Input: c: user-defined number of clusters,
       x1, x2...: the features of state space
       A: set of actions,
Output: learned Q-Map, set of rules

Initialize Q-Map with random values;
Q-Learning_Module()
{
    Repeat {
       randomize the start state s_t;
    // each episode
    Repeat {
     // to select an action at s_t
     generate random number r (0<r<1);
     if (r < threshold1) then exploration;
    else if (Exist_Rule( s_t ))                    //exploitation
          a_t =Recall_Rule_Action( s_t ); // at s_t, action with maximal Q-value is selected
    else a_t=Recall_Q_Map( s_t ) // if there is not any matching rule in the rule set,
                                 an agent searches suitable cluster in Q-Map
    take the a_t and observe next state s_{t+1} and reward;

    // update Q-value of a s_t- a_t pair
         Q(s_t, a_t) = Recall_Q_Map( s_t, a_t );
         max_a Q(s_{t+1}, a_{t+1})=Recall_max( s_{t+1} );   // at s_{t+1}, maximal Q-value is
selected
         update Q(s_t, a_t);

   // rule extraction and management
     if ( s_{t+1} is goal state or reward is penalty) {
               If (Not Exist_Rule( s_t ))       Extraction_Rule( s_t, a_t, reward);
               else Update_Rule( s_t, a_t , Q(s_t, a_t));
     }
    // update Q-Map
     else Update_Q_Map( s_t, a_t, reward, Q(s_t, a_t));
        s_t= s_{t+1};
       } Until( s_t is goal state)
} Until((old Q-value - new Q-value) < threshold2)
```

그림 4 Q-Map 알고리즘

E	S	S	S	E	S	S	S	S	S
E	S	W	E	E	E	E	S	S	W
E	S	O	O	O	O	O	S	S	S
E	S	O	E	E	S	O	S	S	S
E	S	O	E	E	S	O	S	W	W
E	S	O	S	E	G	W	W	W	W
E	E	E	E	E	N	O	N	W	W
E	E	E	N	O	O	O	E	N	W
E	E	N	N	W	W	E	N	N	W
N	N	N	N	N	W	W	N	N	W

그림 5 격자공간 1

S	S	S	S	S	S	S	S	S	W
E	E	E	E	E	E	E	S	S	W
N	N	O	O	O	O	O	E	S	W
E	N	O	E	S	S	S	S	S	W
E	S	O	E	E	S	W	W	W	W
E	S	O	E	E	G	W	W	W	W
E	S	O	O	N	N	N	W	W	W
S	S	O	S	O	O	O	N	N	W
E	W	E	E	E	E	E	N	N	W
N	N	E	N	E	N	N	N	N	N

그림 6 격자공간 2

치명적이므로 규칙으로 추출하고 계속해서 Q값을 관리하게 된다. 학습을 완료하면 다음과 같은 규칙 If (x=7, y=3) Then N=-90.00000, E=-999.00000, S=-3.09258, W=-90.00000)을 얻게 된다. 즉 x=7, y=3의 상태에서는 Q값이 가장 큰 행동 S가 최적임을 나타낸다. 그리고 N(north), E(east), W(west)는 부정적인 행동이므로 매우 낮은 Q값을 갖고 있다. 그림 7은 격자공간 2에서 Q-학습을 수행하면서 추출된 규칙들의 몇 가지예이다.

rule 1: If	9,	7	*Then*	0.22003	-1.82100	-999.00000	-2.29365
rule 2: If	7,	9	*Then*	-1.03406	-999.00000	-1.81100	-0.10330
rule 3: If	7,	7	*Then*	10.20747	-1.05658	-1.05103	-999.00000
rule 4: If	6,	6	*Then*	9.88769	-0.93604	-999.00000	-0.99000
rule 5: If	4,	3	*Then*	-1.50560	-0.10835	-0.90401	-999.00000
rule 6: If	3,	6	*Then*	-999.00000	-1.04132	10.27679	-0.99080
rule 7: If	5,	6	*Then*	0.00000	0.00000	0.00000	11.00618
rule 8: If	6,	4	*Then*	63.78040	-0.99000	-999.00000	-90.00000
rule 9: If	8,	6	*Then*	-999.00000	0.46078	-2.51833	-2.31082
rule 10:If	9,	8	*Then*	-0.16666	-1.71979	-999.00000	-1.71974

그림 7 격자 공간 2의 규칙집합

(x=5, y=6)상태에서 W행동을 취해 보았을 때, 목표로 갈 수 있는 결정적인 행동이었기 때문에 일단 rule 7으로 추출되고, rule 7을 계속해서 학습과정에 이용하면 (x=5, y=6)상태를 포함한 에피소드들은 자연스럽게 목표상태로 유도된다.

표 1은 미로환경 2에서 Q-학습을 수행한 결과를 담고 있는 Q-Map이다. Q-Map의 각 클러스터는 중복 없이 고르게 환경의 각 부분을 계속 적응하면서 관리하고, 학습이 완료되면 요약된 지식을 얻게 된다.

4.2 마운틴 카 문제(mountain car problem)

마운틴 카 문제는 그림 8과 같이 자동차가 목표(goal)에 도달하기 위하여 적절한 가속을 얻도록 관성(coast), 전진(forward), 후진(backward)의 행동을 학습하는 것이다. 마운틴 카 문제에서 상태공간을 구성하는 요소는 위치(P)와 속도(V)이고, 각각 일정한 범위에 속하는 연속값을 갖는다. 위치(P)는 $-1.2 \le P \le 0.5$ 범위내의 값이

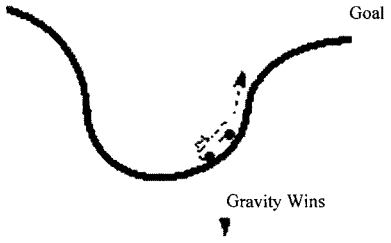


그림 8 마운틴 카 문제

표 1 격자 공간 2의 Q-Map

Cluster No	x	y	Q-value			
			N	E	S	W
1	1.000061	1.002356	-3.664911	-3.664860	-3.664860	-3.665031
2	1.000000	5.009802	-0.036792	-0.046104	-0.018155	-0.017879
3	2.032021	7.707334	-0.164430	-0.230690	2.377735	-0.018655
4	3.990150	0.990149	-0.030800	-0.013459	-0.013479	-0.009000
5	4.188040	4.176032	-0.039107	1.022727	0.393820	-0.051501
6	4.597246	7.106746	1.322052	2.221149	3.276393	11.278080
7	7.999986	1.000000	-2.197308	-2.197308	-2.197308	-2.197308
8	8.000065	3.004346	-2.121330	-2.121330	-2.121330	-2.121330
9	7.852126	7.263744	1.514719	-0.143213	-0.170394	-0.030539
10	5.000000	5.000000				

고, 속도(V)는 $-0.07 \le V \le 0.07$ 사이의 값을 갖는다. 자동차는 동력이 약한 엔진과 중력의 제약조건을 이기고 목표지점인 0.5에 도달하여야 한다.

그림 9는 마운틴 카 문제의 각 요소(feature)를 일정한 간격으로 세밀하게 나누어 얻어진 952개의 상태에 대하여 룩업테이블을 이용하여 Q-학습을 수행한 결과와 100개의 클러스터로 구성된 Q-Map을 이용한 Q-학습의 결과를 비교한 그래프이다. 그래프의 x축은 시도하는 상태-행동 쌍의 수를 표시하고, y축은 테스트 집합의 1000개의 상태 중에서 목표에 도달하는 상태의 수를 나타낸다. 데이타가 연속값을 갖는 문제에 기본적인 Q-학습을 수행하는 경우, 상태공간을 일정한 간격으로 세밀하게 나누더라도 중심값이 경험에 따라 이동하는 Q-Map과 비교하여 학습능력이 떨어짐을 보인다. 시도횟수가 50만 번 이하에서 기복이 심한 이유는 상태공간을 충분히 탐색하지 못했음을 의미한다. 그리고 5백만번 이상이면 오버핏팅(over-fitting)이 된다.

그림 10은 초기에 랜덤하게 결정된 클러스터의 중심값과 Q-Map을 이용하여 적응된 중심값을 나타낸다. 중심값을 다양하게 초기화 하여 실험한 결과, 적응된 중심값들의 위치는 항상 그림 10과 같았다.

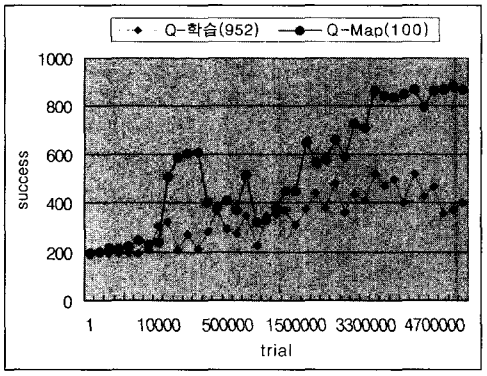


그림 9 기본적인 Q-학습 적응형인 Q-Map 비교

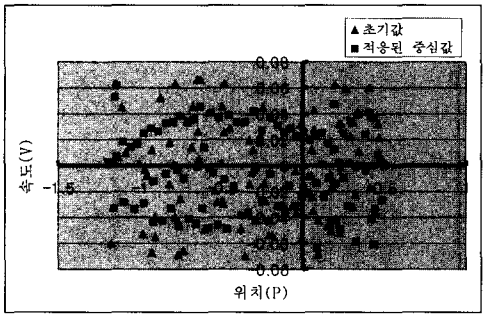


그림 10 초기의 중심값과 적응된 중심값

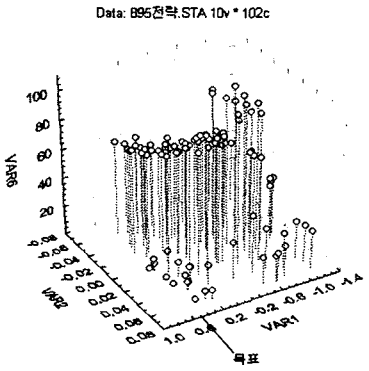


그림 11 Q-Map을 이용한 이동횟수

그림 11은 각 위치 VAR1에서 VAR2의 속도일 때, 학습된 Q-Map을 이용해서 목표(위치=0.5)까지 도달하기 위하여 필요한 이동횟수(VAR6)를 나타낸 그래프이다. 현재 자동차의 위치가 목표지점에 가깝더라도 속도가 0미만이면, 전진으로 목표에 도달하지 못하므로 가속을 얻기 위해서 후진(backward)을 하기 때문에 이동회수가 40 이상이다. 그리고 위치가 0 미만은 목표지점에서 멀지만 속도가 0.04이상으로 가속되어 있다면 적은 수의 이동으로 목표에 도달함을 보인다.

5. 결론과 향후 연구과제

본 논문에서 제안한 Q-Map은 문제(task)에 따라 상태공간이 거대해지는 문제점을 해결하기 위하여 상태공간에서 지역적으로 유사한 상태들을 클러스터링 하였다. Q-Map의 각 클러스터는 중심값과 행동 전략을 기억하며 계속해서 새로운 경험에 적응해 나간다. 각 클러스터의 중심값은 규칙의 조건부에 해당하므로, 별도의 복잡한 규칙 추출 과정이 없이 직접적으로 추출을 할 수 있다. Q-Map의 중심값들은 적응하는 과정에서 조금씩 이동하므로, 이산 데이타를 갖는 문제보다는 연속데이타를 갖는 제어문제에서 기존의 Q-학습보다 높은 학습능력을 보였다.

향후 연구과제는 다음과 같다. 첫째, 유클리드 거리는 물리적으로 인접한 상태들(physically adjacent states)을 구할 수 있는데, 많은 제어문제(control problems)인 경우 인접한 상태사이의 전이가 불가능한 경우가 많다. 그러므로 값의 유사성이 아닌 "가깝다"의 다른 개념 정의가 필요하다[9,10]. 둘째, Q-Map을 이용한 학습은 기본적인 Q-학습과 마찬가지로, 현재 경험한 상태-행동 쌍의 Q값만을 갱신하므로 학습 속도가 느리다. 입력 상태가 각 클러스터에 속하는 정도를 표시하는 퍼지 논리의 소속도를 이용하여 속도 개선을 연구할 계획이다.

참 고 문 헌

[1] Stuart I. Reynolds, *Adaptive Resolution Model-Free Reinforcement Learning: Decision Boundary Partition*, Advances in Artificial Intelligence, 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence(AI-2001), Ottawa, Canada, June 2001, Proceedings.

[4] Michael Herrmann, Ralf Der, *Efficient Q-Learning by Division of Labor*, in Proc. International Conference on Artificial Neural Networks-ICANN'95, Vol. II, S.129-134.

[5] Ron Sun, *knowledge Extraction from Reinforcement Learning*, Proceedings of International Joint Conference on Neural Networks, Washington, DC. July 10-15, 1999. IEEE Press, Piscataway, NJ.

[6] Rudy Setiono and Huan Liu, *Symbolic Representation of Neural Networks*, IEEE Computer March 1996 (Vol. 29, No. 3) pp. 71-77.

[7] Ron Sun, *Supplementing Neural Reinforcement Learning with Symbolic Methods: Possibilities and Challenges*, Proceedings of International Joint Conference on Neural Networks, Washington, DC. July 10-15, 1999. IEEE Press, Piscataway, NJ.

[2] Richard S. Sutton, *Generalization in Reinforcement Learning: Successful Examples Using sparse Coarse Coding*, Advances in Neural Information Processing Systems, pp.1038-1044, MIT Press, 1996.

[3] Edward Keedwell, Ajit Narayanan and Dragon Savic, *Using Genetic algorithms to extract rules from trained neural networks*, Proceedings of the Genetic and Evolutionary Computing Conference, Volume 1, Morgan Kaufmann Publishers, San Francisco, California, USA, 1999: 793.

[8] R.Matthew Kretchmar, Charles W. Anderson, *Comparison of CMACs and Radial Basis Functions for Local Function Approximators in Reinforcement Learning*, ICNN'97. International Conference on Neural Networks. 1997.

[9] Haixun Wang, Wei Wang, Jiong Yang, Philip S. Yu, *Clustering by Pattern Similarity in Large Data Sets*, ACM SIGMOD Conference 2002 Madison, Wisconsin, USA.

[10] R. Matthew Kretchmar, Charles W. Anderson, *Using Temporal Neighborhoods to Adapt Function Approximators in Reinforcement Learning*, IWANN99: International Work Conference on Artificial and Natural Neural Networks : Alicante, Spain. June 1999.

[11] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA., 1998.

이 영 아

정보과학회논문지 : 소프트웨어 및 응용
제 30 권 제 9 호 참조

홍 석 미

정보과학회논문지 : 소프트웨어 및 응용
제 30 권 제 9 호 참조

정 태 충

정보과학회논문지 : 소프트웨어 및 응용
제 30 권 제 9 호 참조