인공지능

Project2

담당 교수님: 박철수

강의 시간: 금3,4

학 과: 컴퓨터정보공학부

학 번: 2020202055

성 명: 최소윤

1. Introduction

Dynamic programming을 이용하여 environment의 model을 푸는 과제로 큰 순차적인 행동 결정을 작은 process로 나누어 제시된 grid world에서 Policy iteration과 Value iteration을 policy를 최적화하는 것이 목표이다. 7 x 7의 시작점과 끝점을 가지는 grid-world로 action은 상하좌우이다. 이번 프로젝트에서 Policy evaluation과 Policy Improvement, Value Iteration을 구현해보도록 한다.

1. Algorithm

**Policy evaluation – random Policy**

Matrix가 주어지면 각 state에 대한 ture value function을 찾는 것이 목표이며 이는 현재 따르는 policy가 맞는지 틀린지 판단하기 위해 value function을 사용하며 value function을 계속해서 update하여 true value function을 찾는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

State의 k+1번째 value function은 state에서 이동 가능한 다음 state에 대해서 k번째 구해놓은 다음 state의 value function을 이용한다.

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

위 그림과 같이 초기 value function을 0으로 설정하고 state에서 현재 V0에서 one-step씩 이동 가능한 모든 다음 state의 value function으로 다음 iteration의 value function(V1)을 구하여 update할 수 있다. 위, 아래, 왼쪽, 오른쪽 각 행동에 대한 action이 가능하며 각 action에 대한 value function을 구한 뒤 더한 값을 저장한다. 위 행동을 반복하면 수렴하는 값이 나오게 되는데 이는 ture value function이 된다.

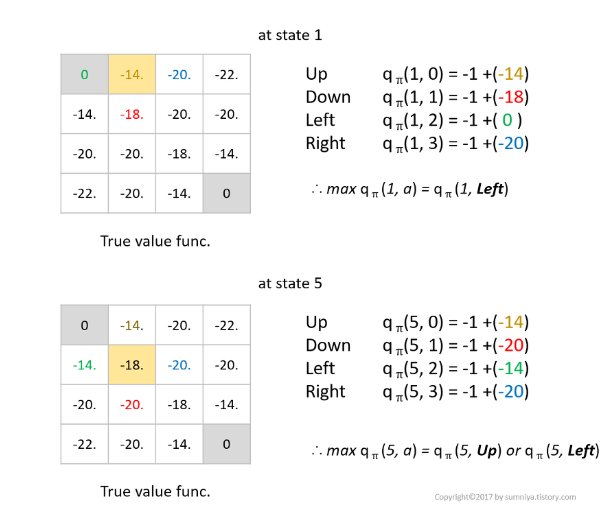
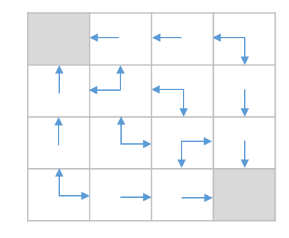
**Policy Improvement**

Iteration을 반복하여 true value function을 찾은 후에 해당 policy를 따르는 것이 좋은 것인지 판단하고 policy를 update해야한다. 이 과정을 통해 optimal policy를 찾을 수 있다. Greedy policy improvement는 가장 널리 알려진 방법으로, 가치가 가장 높은 state를 선택하겠다는 의미로 max값을 가져온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

policy evaluation에서 state-value function을 이용하여 모든 state에 대한 value값을 구했으면 policy에 따른 action을 취해야한다. 현재 state에서 갈 수 있는 가장 높은 state로 이동하는 action을 취해야 하며 action을 정량화하여 선택하는 기준을 만드는 q-function을 이용해야 한다.

state별로 취할 수 있는 action에 대해 q-function을 구한다. 이는 greedy policy improvement를 통해 선택된 action이며 이를 모든 state에 대해 적용하면 오른쪽 그림과 같이 화살표로 action을 표현할 수 있으며 optimal policy를 찾게 된다.

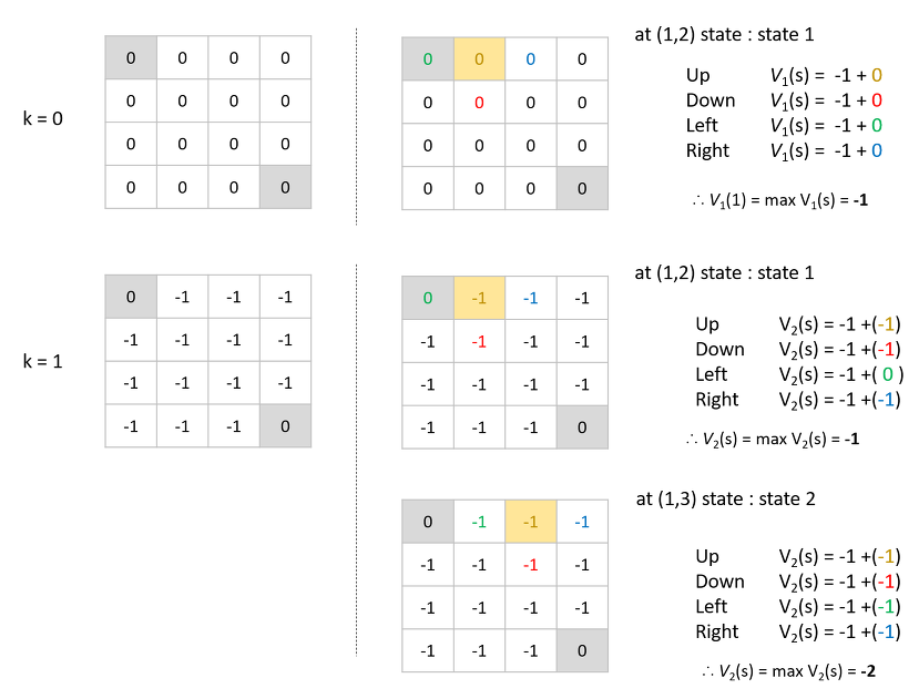
**Value Iteration**

Policy Iteration과는 다르게 Bellman Optimality Equation을 사용한다. 이는 evaluation을 한 번만 진행하며 이동 가능한 다음 state에 대한 모든 value function 중에서 max값만 가져와 greedy하게 value function을 구한 후 improve하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 식은 max값을 취하는 optimal value function이다.



위 그림을 보면 각 취할 수 있는 action에 대해 다음 state에 대한 value값을 모두 더하는 것이 아니라 최대값을 선택하여 value 값을 취한다. 이를 통해 빠른 수렴이 가능하다.

1. Result

**Policy evaluation – random Policy**

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

상하좌우로 이동할 수 있으며 각 확률은 0.25로 같다. 7 x 7 grid world에서 random policy에 대한 true value function을 구할 수 있었으며 이를 계산하여 iteration을 1000으로 설정하고 결과 값을 출력했을 때 다음 그림과 같이 나왔다. Iteration 0에서와 같이 초기값을 start와 end에서는 0으로 설정하였고 나머지는 -1로 초기화하고 중간에 -100으로 함정을 만들었다. one step씩 각 state의 value function을 update하면 초반에는 값이 점점 작아지면서 값이 계속 변하는 것을 알 수 있다.

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

위와 같이 iteration 1000번째에는 수렴한 값이 나오는 것을 확인할 수 있으며 iteration 711번째에서 값이 수렴하였다.

**Policy Improvement**

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

Policy evaluation을 통해 iteration을 반복하여 true value function을 찾았고, Policy Improvement를 통해 어떤 policy를 따르는 것이 좋을지 안 좋을지 판단하여 알맞은 Policy를 update하여 이를 반영해야 한다. 현재 policy보다 더 나은 policy를 찾아가면 optimal policy에 가까워진다.

테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명

Greedy policy improvement로 구현하여 max값만 선택한다. 그리고 해당 policy에 따른 action을 취해야한다. 즉 현재 state에서 가장 높은 곳으로 이동하는 action을 취한다. 위의 그림은 update된 policy를 의미한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 그림은 각각의 state에서의 action을 matrix로 나타낸 모습이다.

**Value Iteration**

테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 Value Iteration을 적용한 것이며 이는 Bellman Optimally equation을 이용하여 계산한다. Evaluation 과정에서 max값을 취해서 greedy하게 value function을 구한다. Policy evaluation과 비교하면 Value Iteration은 Iteration 5번째부터 수렴했으며 더 빨리 수렴한 것을 볼 수 있다.

**Random policy & Greedy policy**

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

random policy에서는 첫번째 그림과 같이 수렴하는 true value가 나왔으며, Policy Improvement를 구현하여 greedy policy에서는 두번째 그림과 같이 optimal policy를 도출할 수 있었다. 결과적으로는 policy Improvement를 greedy policy로 구현하여 optimal한 action을 찾을 수 있었으며 첫번째 그림의 true value의 최대값에 따라 action을 결정하면 두번째 그림과 같이 action이 나온다. 그러므로 결국 optimal policy는 두번째 그림과 같이 action을 취하면 reward를 최대로 받아야하는 agent는 최단거리로 종료점까지 갈 수 있다.

1. Consideration

Policy evaluation과 Policy Improvement와 Value Iteration 용어와 개념이 헷갈려 이해하는 데 어려움이 있었다. 이번 과제를 통해서 강의 자료를 복습하면서 이해를 하고 프로젝트를 진행하면서 코드로 구현하고, 한 줄씩 이해해보는 과정을 통해 완전히 이해할 수 있었다. 또한 공식도 이해하기 어려웠는데 한 칸씩 계산해보면서 해당 value가 어떻게 나왔는지 직접 계산해보았고 이를 통해 코드를 작성할 때 좀 더 수월하게 작성할 수 있었다.

