**Homework 2**

2018324133 김태우

이 문제의 해결을 위해 MDP에서의 Value Iteration 방법을 사용하였다. state는 현재 재산(0~100)이며 action은 배팅한 금액 (0~min(s, (100 - s))), reward는 state가 100일 경우에만 1이다.

Value Function v는 이다.

Undiscounted, Finite, Episodic MDP이므로 γ=1이며, action에 따른 state는 2개(win, lose)이므로 식을 다시 쓰면,

가 된다.

본 문제의 해결을 위해 Python과 numpy 라이브러리를 사용하였다.

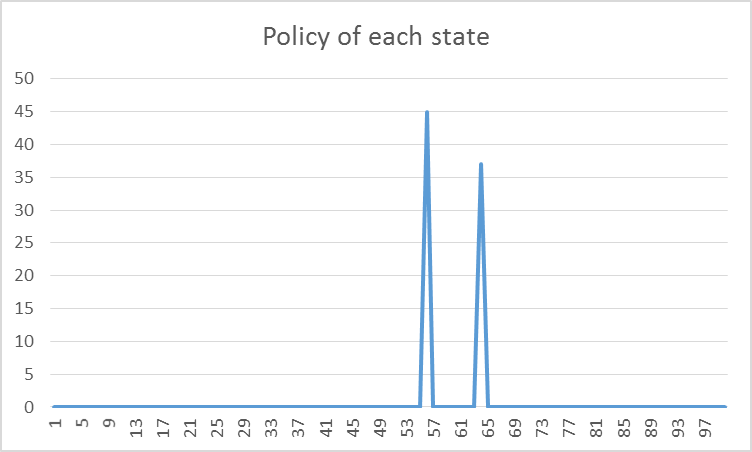
|  |
| --- |
| P\_HEAD = 0.4  GAMMA = 1.0  # initial values of states are 0  values = [0.] \* 101  # the rewards of state 0-99 are 0  rewards = [0.] \* 101  # except the final state  rewards[100] = 1.  # initial policy of state 0-99 are 0  policy = [0] \* 100 |

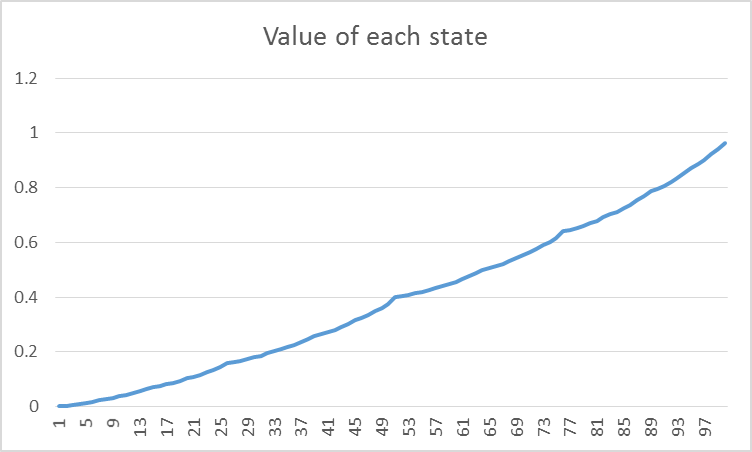
우선 상수들과 초기값을 정의하였다.

|  |
| --- |
| delta = 100000  iter = 0  while delta > 1e-20 : # the value is optimal therefore no changes occured  iter += 1  delta = 0.  newvalues = [0] \* 100  for it\_state in range(100) :  for it\_action in range(min(it\_state, 100 - it\_state) + 1) :  newval =P\_HEAD \* (rewards[it\_state + it\_action] + \  GAMMA \* values[it\_state + it\_action]) +\  (1. - P\_HEAD) \* (rewards[it\_state - it\_action] + \  GAMMA \* values[it\_state - it\_action])  if newval > newvalues[it\_state] :  newvalues[it\_state] = newval  policy[it\_state] = it\_action  if delta < (abs(values[it\_state] - newvalues[it\_state])) :  delta = abs(values[it\_state] - newvalues[it\_state])  values[:100] = newvalues[:] |

그리고 value가 안정화될때까지 계속 루프를 돌도록 하였다. 각 Iteration마다 각 state의 새로운 value를 구해서 반영하고, 기존 value와 새 value의 차이값이 1e-15보다 작아질 때 까지 반복한다.

그런데 이 식을 돌려본 결과



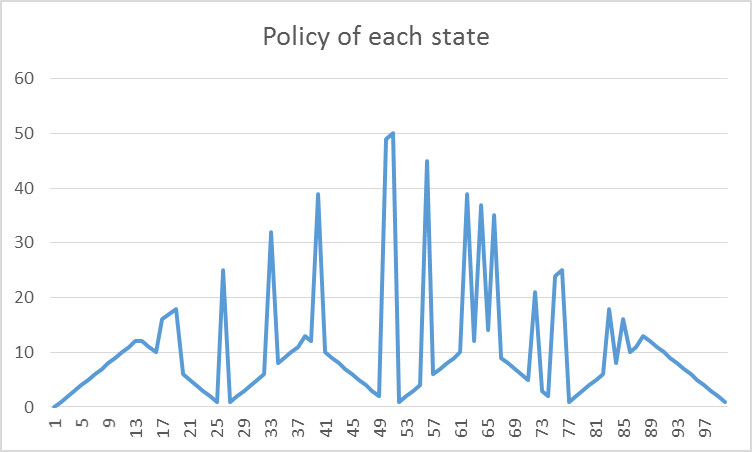


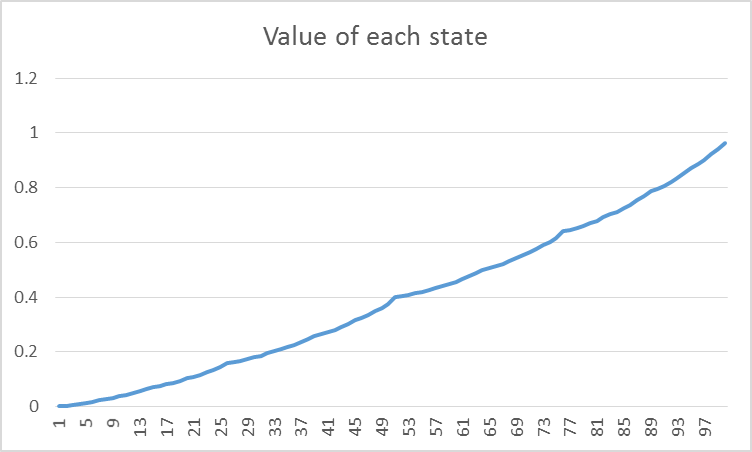
Value는 어느 정도 예시의 그래프와 유사한데 Policy가 거의다 0으로 수렴해 버림을 알 수 있었다.

Policy가 0이 되는건 사실 본 문제는 이익을 얻을 확률이 40%밖에 되지 않는 도박이므로 하지 않는것이 가장 유리하다는 당연한 결과로도 볼 수 있으나, 0달러만 계속 걸면 state가 유지되어 끝나지 않게 되므로 우리가 원했던 답은 아닐 것이다. 따라서 action을 최소 1달러는 걸도록 제한하였다.

|  |
| --- |
| ...  iter += 1  delta = 0.  newvalues = [0] \* 100  for it\_state in range(100) :  for it\_action in range(1, min(it\_state, 100 - it\_state) + 1) :  ... |

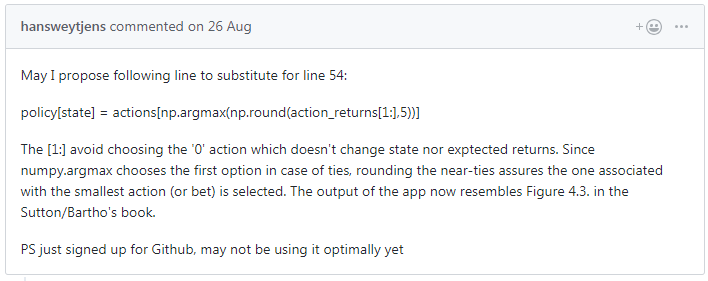
이렇게 할 경우의 결과는 다음과 같다.





Value는 거의 바뀌지 않았지만 Policy가 상당히 난해하게 수렴했음을 확인할 수 있었다. 알고리즘이 예시와 다른 결과를 내놓은 이유를 알기 위해 검색을 해본 결과, 다음을 찾을 수 있었다.

https://github.com/ShangtongZhang/reinforcement-learning-an-introduction/issues/83



action을 구할 때, 0달러를 거는 action은 고려해서는 안된다는 위에서 발견한 사실과 더불어, 거의 같은 value일 경우 더 금액이 작은 action을 policy로 취하는 것을 제안하고 있다. 이를 위해 value를 0.00001 단위에서 반올림한 후, 같으면 더 작은 쪽을 취하면 되고, 이 경우 sutton 교수의 그래프와 같은 결과가 나온다고 한다.

|  |
| --- |
| ...  newval =P\_HEAD \* (rewards[it\_state + it\_action] + \  GAMMA \* values[it\_state + it\_action]) +\  (1. - P\_HEAD) \* (rewards[it\_state - it\_action] + \  GAMMA \* values[it\_state - it\_action])  if np.round(newval, 5) > np.round(newvalues[it\_state], 5) :  newvalues[it\_state] = newval  policy[it\_state] = it\_action  ... |

실제로 policy를 구하는 과정은 제안한 대로 변경해 본 결과는 다음과 같았다.

Policy가 거의 예시와 비슷하게 나왔음을 확인 할 수 있었다.