CS7641 Assignment 4 - Eric Gregori (04/22/18)

Interesting MDP's

A Markov Decision Process (MDP) is a tuple containing: a finite set of states, a finite set of actions, a state transition probability function, a reward function, and a discount factor. [1] This assignment will analyze two MDP's in the form of grid worlds with obstacles.

MDP 4 x 4 Grid World

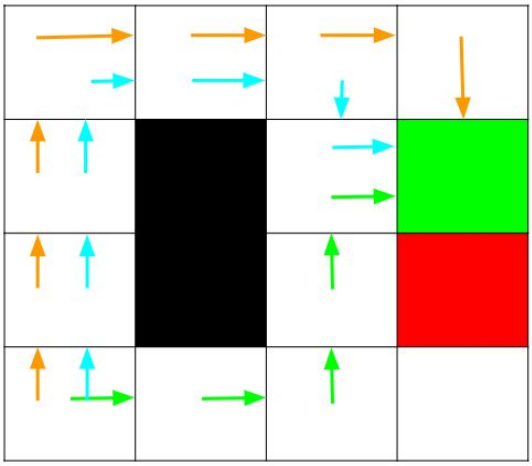


Figure 1. Small MDP (14 states)

MDP 11 x 11 Grid World

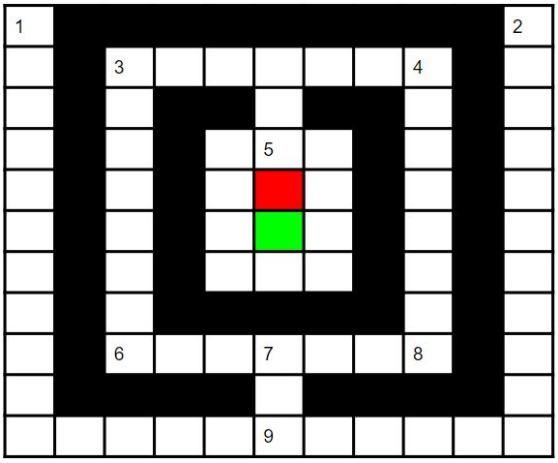


Figure 2. Large MDP (71 states)

The small MDP (figure 1) is a 4x4 grid world containing 14 states with three potential paths. The shortest path (green) is 5 transitions while the other two paths (orange, blue) are 7 transitions. The large MDP (figure 2) is a 11x11 grid world with 71 states. The numbers in figure 2 represent "**interesting" states**. States 1 and 2 are interesting because they have no value. State 5 and 7 are interesting because they represent an equal decision (left == right). **How do the algorithms decide which way to go?** Finally, state 9 makes sense for an algorithm working from the terminal state back, but how would an algorithm starting from the lower left corner approach the state 9 decision.

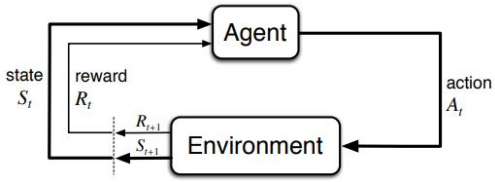


Figure 3. from Sutton and Barto, Reinforcement Learning: An Introduction, 1998

MDP Domain Knowledge	Description
reward (terminal)	A positive (green) or negative (red).
reward	The default reward for each non terminal state.
probability of action	This is the probability that the desired action is taken.
Initial state (S0)	Lower left corner by default.

As shown in figure 3, MDP domain knowledge is encoded in the transition model and rewards. "In an MDP, the transition function and reward function do not change. This means that an MDP behaves according to the Markov assumption, which states that the probabilities of going to a state s ′ after performing an action a in state s only depend on s and a, not on the history of the environment before the agent encountered s – there is no hidden information." [5]

Solving Using Value Iteration

Then solution $v_*(s')$ can be found by one-step lookahead

$$v_*(s) \leftarrow \max_{a \in \mathcal{A}} r(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s'|s,a) v_*(s')$$

The idea of value iteration is to apply these updates iteratively

Intuition: start with final rewards and work backwards

"Value Functions are Cumulative Expected Rewards.Optimal Value Functions are Best Achievable Cumulative Expected Rewards." [1]

< Figure 4. Value Iteration [2]

Experiment 1 - Solve small MDP using Value Iteration

Small MDP, reward: 0, terminal reward: +-1.0, probability: 0.1,0.8,0.1, S0:(0,0), γ: 0.9

		Iteration 2						Iteration 3						Iteration 4			
S	A	P(S'|S,A)	S1	U[S']	P*U[S']	S	A	P(S'|S,A)	S1	U[S']	P*U[S']	S	A	P(S'|S,A)	S1	U[S']	P*U[S']
(3, 3)	R	0.8	(3, 3)	0	0	(3, 3)	R	0.8	(3, 3)	0.72	0.576	(3, 3)	R	0.8	(3, 3)	0.7848	0.62784
(3, 3)	R	0.1	(3, 2)	1	0.1	(3, 3)	R	0.1	(3, 2)	1	0.1	(3, 3)	R	0.1	(3, 2)	1	0.1
(3, 3)	R	0.1	(3, 3)	0	0	(3, 3)	R	0.1	(3, 3)	0.72	0.072	(3, 3)	R	0.1	(3, 3)	0.7848	0.07848
(3, 3)	U	0.8	(3, 3)	0	0	(3, 3)	U	0.8	(3, 3)	0.72	0.576	(3, 3)	U	0.8	(3, 3)	0.7848	0.62784
(3, 3)	U	0.1	(3, 3)	0	0	(3, 3)	U	0.1	(3, 3)	0.72	0.072	(3, 3)	U	0.1	(3, 3)	0.7848	0.07848
(3, 3)	U	0.1	(2, 3)	0	0	(3, 3)	U	0.1	(2, 3)	0	0	(3, 3)	U	0.1	(2, 3)	0.5832	0.05832
(3, 3)	L	0.8	(2, 3)	0	0	(3, 3)	L	0.8	(2, 3)	0	0	(3, 3)	L	0.8	(2, 3)	0.5832	0.46656
(3, 3)	L	0.1	(3, 3)	0	0	(3, 3)	L	0.1	(3, 3)	0.72	0.072	(3, 3)	L	0.1	(3, 3)	0.7848	0.07848
(3, 3)	L	0.1	(3, 2)	1	0.1	(3, 3)	L	0.1	(3, 2)	1	0.1	(3, 3)	L	0.1	(3, 2)	1	0.1
(3, 3)	D	0.8	(3, 2)	1	0.8	(3, 3)	D	0.8	(3, 2)	1	0.8	(3, 3)	D	0.8	(3, 2)	1	0.8
(3, 3)	D	0.1	(2, 3)	0	0	(3, 3)	D	0.1	(2, 3)	0	0	(3, 3)	D	0.1	(2, 3)	0.5832	0.05832
(3, 3)	D	0.1	(3, 3)	0	0	(3, 3)	D	0.1	(3, 3)	0.72	0.072	(3, 3)	D	0.1	(3, 3)	0.7848	0.07848
Max		0.8	0.9	**0.72**		Max		0.872	0.9	**0.7848**		Max		0.9368	0.9	**0.8431**	

Figure 5. U(3,3) (upper right box) value calculations

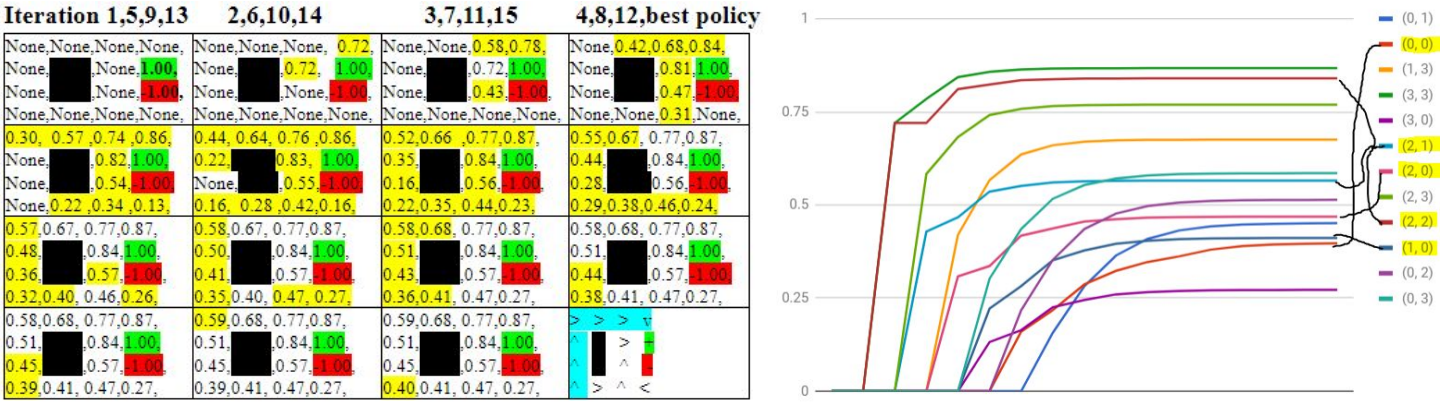


Figure 6. Solving small MDP using value iteration (reward = 0)

Analysis

The value iteration algorithm does not take into account "start or terminal" states. The algorithm simply smooths the reward from the states with high reward to states with low reward. As shown in figure 6, the algorithm flows like water from states with high reward to neighboring states with lower reward.

Figure 5 shows the calculation details for state (3,3) (upper right corner). The calculations show how state (3,3) absorbs the reward from the terminal state (3,2) in the first iteration, absorbs award from state (3,2) and itself in the second iteration, and finally from (3,2), itself, and (2,3) in the third iteration. As shown in figure 6, state (3,3) continues to increment up to state 7.

The final best policy is shown in figure 6 (highlighted in blue). **Note, the algorithm did not find the shortest route. Taking 7 instead of 5 steps to get to the terminal state. Is this because there was no penalty for taking a longer route? (SEE EXPERIMENT 5)**

Experiment 2 - Impact of reward on small MDP value iteration

Small MDP, reward: -0.04, terminal reward: +-1.0, probability: 0.1,0.8,0.1, S0:(0,0), γ: 0.9

Iteration 1,5,9,13	2,6,10,14	3,7,11,15	4,8,12,best policy
-0.04,-0.04,-0.04,-0.04, -0.04,None,-0.04,1.00, -0.04,None,-0.04,-1.00, -0.04,-0.04,-0.04,-0.04,	-0.08,-0.08,-0.08,0.67, -0.08,None,0.67,1.00, -0.08,None,-0.08,-1.00, -0.08,-0.08,-0.08,-0.08,	-0.11,-0.11,0.50,0.73, -0.11,None,0.67,1.00, -0.11,None,0.35,-1.00, -0.11,-0.11,-0.11,-0.11,	-0.14,0.30,0.59,0.79, -0.14,None,0.76,1.00, -0.14,None,0.38,-1.00, -0.14,-0.14,0.19,-0.14,
0.15,0.44,0.65,0.80, -0.16,None,0.77,1.00, -0.16,None,0.45,-1.00, -0.16,0.07,0.21,-0.01,	0.28,0.51,0.67,0.81, 0.04,None,0.78,1.00, -0.19,None,0.46,-1.00, -0.02,0.12,0.29,0.02,	0.35,0.53,0.67,0.81, 0.17,None,0.78,1.00, -0.05,None,0.47,-1.00, 0.03,0.19,0.31,0.08,	0.39,0.54,0.68,0.81, 0.24,None,0.78,1.00, 0.07,None,0.48,-1.00, 0.10,0.21,0.32,0.10,
0.41,0.54,0.68,0.81, 0.28,None,0.78,1.00, 0.15,None,0.48,-1.00, 0.13,0.23,0.33,0.11,	0.41,0.55,0.68,0.81, 0.30,None,0.78,1.00, 0.19,None,0.48,-1.00, 0.15,0.24,0.33,0.12,	0.42,0.55,0.68,0.81, 0.31,None,0.78,1.00, 0.21,None,0.48,-1.00, 0.16,0.24,0.34,0.12,	0.42,0.55,0.68,0.81, 0.32,None,0.78,1.00, 0.22,None,0.48,-1.00, 0.17,0.25,0.34,0.12,
0.42,0.55,0.68,0.81, 0.32,None,0.78,1.00, 0.23,None,0.48,-1.00, 0.17,0.25,0.34,0.12,	0.42,0.55,0.68,0.81, 0.32,None,0.78,1.00, 0.23,None,0.48,-1.00, 0.17,0.25,0.34,0.12,	0.42,0.55,0.68,0.81, 0.32,None,0.78,1.00, 0.23,None,0.48,-1.00, 0.17,0.25,0.34,0.12,	> > > v ^ N > + ^ N ^ - > > ^ <

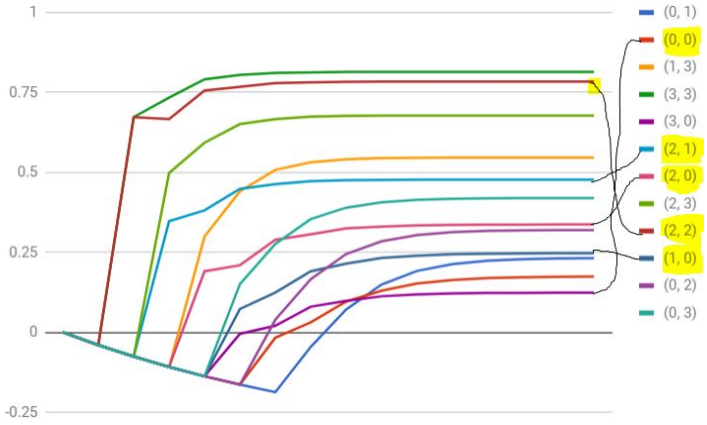


Figure 7. Solving small MDP using value iteration (reward = -0.04)

Analysis

Changing the reward from 0 to -0.04 changed the best policy solution to the shortest path. This policy change is a result of state (0,0) changing from up (reward=0) to right (reward=-0.04). The change in direction is a result of changes in the values in the neighboring states (0,1), (1,0). **With reward=0, (0,1) > (1,0). With reward=-0.04, (0,1)<(1,0).**

As shown in figure 7, with negative reward, the states furthest from the terminal states initially decrease (this can be seen as red). After initially decreasing, they start to increase in later iterations (shown in green). As shown by the graph in figure 7, the states furthest from the terminal state decrease considerably before increasing. State (0,1) is the last to increase in both instances (blue line in figures 6,7). In figure 6 the relative values for states (0,1) and (1,0) swap (the lines cross). In figure 7, the lines do not cross.

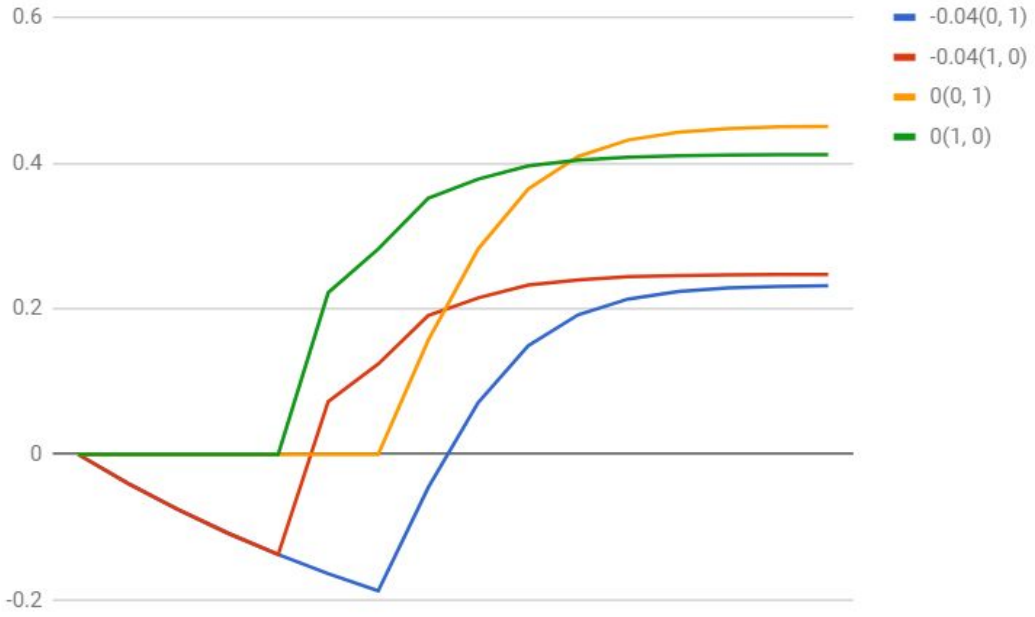


Figure 8. States (0,1) and (1,0) values with rewards 0 and -0.04

Figure 8 shows why the policy changed when the reward changed from 0 to -0.04. **In summary, a negative reward results in states furthest from the terminal state to decrease before increasing in value. The decrease is proportional to the distance from the terminal state. The result is a shortest route policy.**

Experiment 3 - Solve large MDP using Value Iteration

Large MDP, reward: 0, terminal reward: +-1.0, probability: 0.1,0.8,0.1, S0:(0,0), γ: 0.9

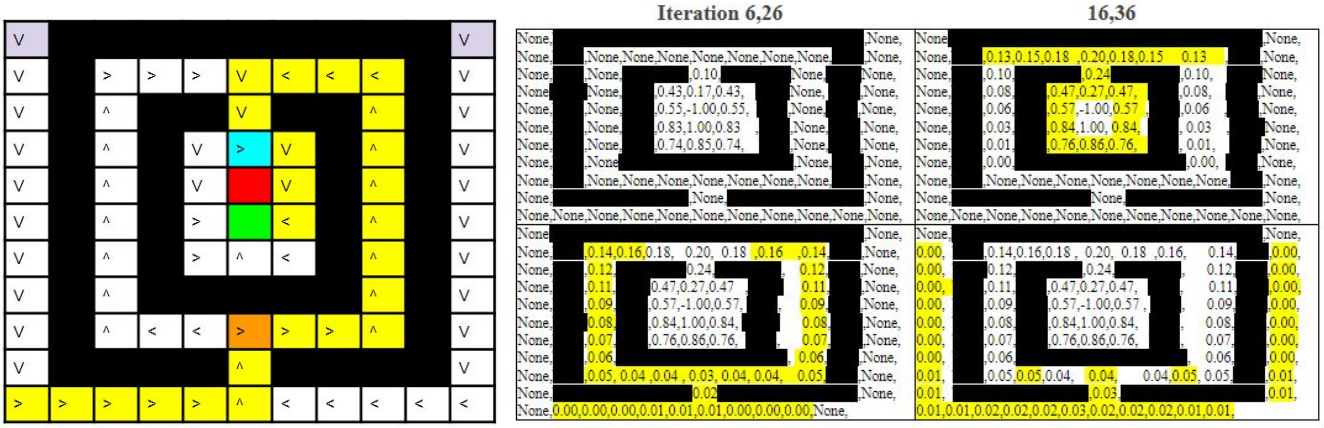


Figure 9. Solving large MDP using value iteration (reward = 0)

Analysis

States (4,2) and (6,2) are on either side of the orange state in figure 9. These values are equal. States (4,7) and (6,7) are on either side of the blue state in figure 9. These values are also equal. **Hypothesis - the algorithm implementation simply selects right when both options are equal.**

Experiment 4 - Impact of reward on large MDP value iteration

Large MDP, reward: -0.04, terminal reward: +-1.0, probability: 0.1,0.8,0.1, S0:(0,0), γ: 0.9

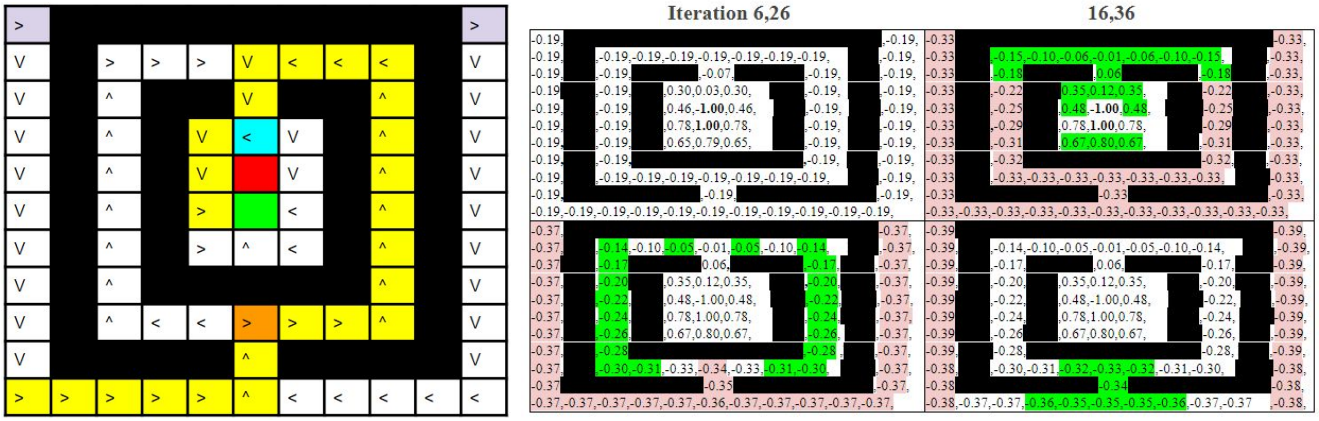


Figure 10. Solving large MDP using value iteration (reward = -0.04)

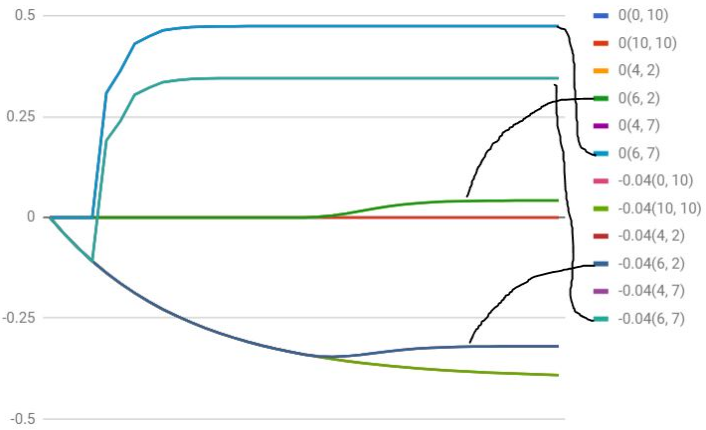


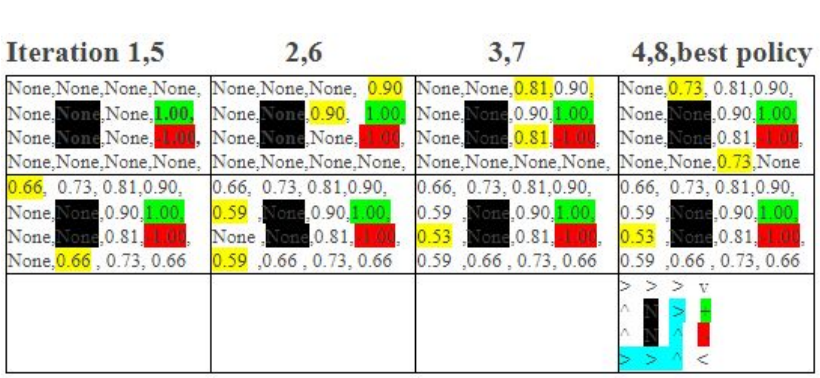
Figure 11. State values on either side of key decision points

Analysis

This experiment proved right selection hypothesis incorrect. The direction decision which equal value neighbors appears to be random. As shown in figure 11, the neighbor states for both the orange and blue state are equal yet the result was different. **The algorithm does not have an explicit rule for what to to do when neighbors have equal values. The result is a random decision.**

Experiment 5 - Impact of probability of action on Value Iteration

Small MDP, reward: 0, terminal reward: +-1.0, probability: 0,1.0,0, S0:(0,0), γ: 0.9



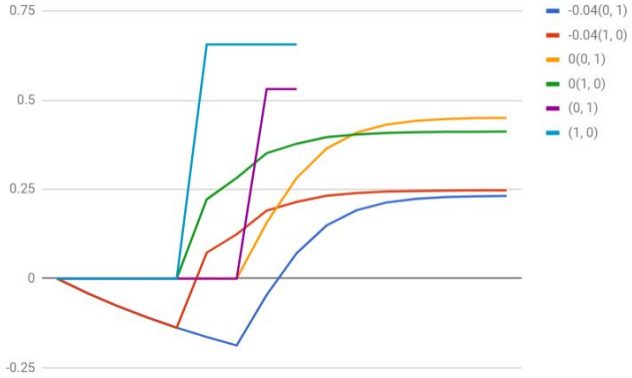


Figure 12. Small MDP with probability 0, 1.0, 0 all other parameters same as figure 6

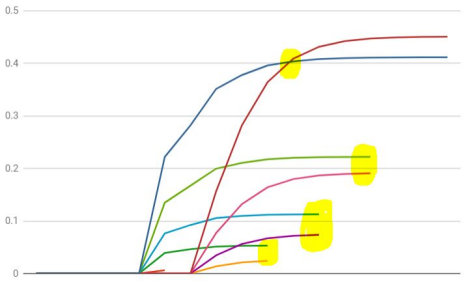
Analysis

As with experiment 1, the reward in this experiment is 0, but the algorithm calculated a different route. In comparing the values between figure 6 and figure 12, note states (0,1) and (1,0). The values do not gradually increase, they go from 0 to their final value. **The probability parameters provide the "smoothing". Connecting the neighboring states over the iterations.**

The algorithm chose the direct route because the -1.00 terminal state was not a factor. In experiment 1, state (1,0) was impacted by the negative terminal state due to the probability function for state (2,1). That value propagated to state (1,0).

Experiment 6 - Impact of gamma on Value Iteration

Small MDP, reward: 0, terminal reward: +-1.0, probability: 0.1,0.8,0.1, S0:(0,0)



< Figure 13. γ = 0.4, 0.6, 0.7, 0.8, 0.9
γ = 0.4 -> 6 iterations - longer route (avoid negative terminal state)
γ = 0.6 -> 10 iterations - longer route (avoid negative terminal state)
γ = 0.7 -> 12 iterations - longer route (avoid negative terminal state)
γ = 0.8 -> 14 iterations - longer route (avoid negative terminal state)
γ = 0.9 -> 15 iterations - shorter route

Analysis

As the gamma increases, the number of iterations increases and the solution changes. Based on the value equation and the result of experiment 5, **as gamma increases the impact of the negative terminal state increases resulting in a policy shift from avoiding the -1.0 state to favoring a shorter route.**

Solving Using Policy Iteration

repeat
  $\pi \leftarrow \pi'$
  $U \leftarrow ValueDetermination(\pi)$; reverse from value iteration
  for each state s do
    $\pi'[s] \leftarrow \arg\max_a \sum_{s'} P(s' \mid s,a)U(s')$
  end
until $\pi = \pi'$

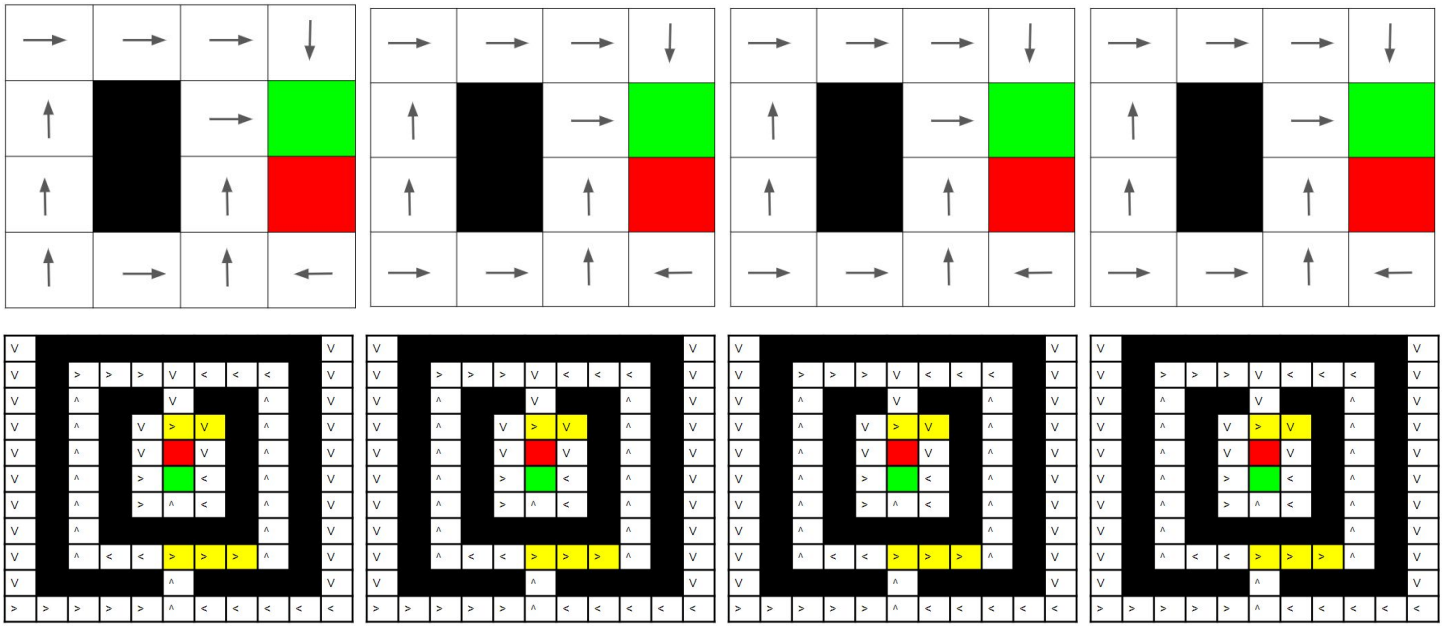
"A policy is a choice of what action to choose at each state An Optimal Policy is a policy where you are always choosing the action that maximizes the "return"/"utility" of the current state. An optimal policy maximizes expected sum of rewards." [4]

< Figure 14 Policy Iteration [4]

Experiment 7 - Impact of domain knowledge on Policy Iteration

terminal reward: +-1.0, S0:(0,0), γ: 0.9

probability: 0.1,0.8,0.1 **probability: 0,1.0,0**



reward: 0 **reward: -0.04** **reward: 0** **reward: -0.04**

Figure 15. Experimenting with domain knowledge and policy iteration

Analysis

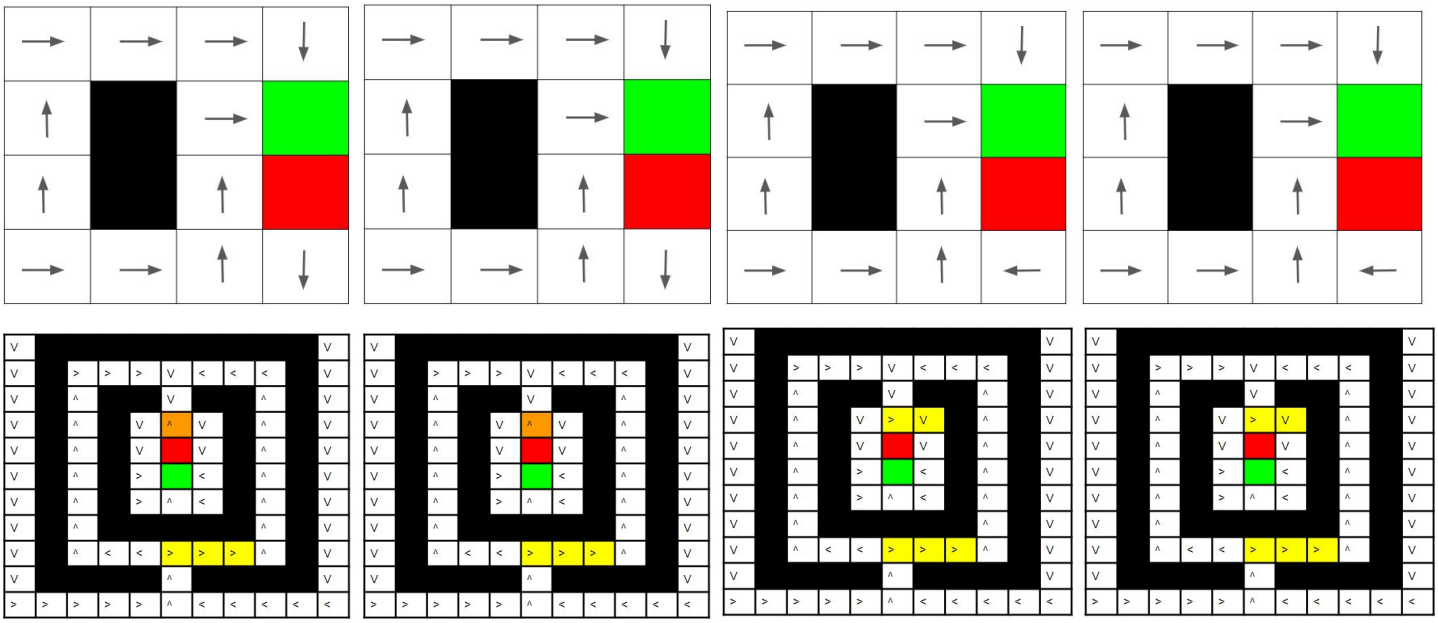
Like value iteration (experiment 2) policy iteration avoids the negative terminal state if there is no penalty for taking a step. If there is chance of turning into the negative terminal state (probability 0,1,0), the shortest path is the best policy. For the small MDP, policy iteration returns the same policies as value iteration. This makes sense because policy iteration uses the same utility function.

For the large MDP, it appears that if the decision is equal, policy iteration always goes right.

Experiment 9 - Impact of gamma on Policy Iteration

reward: 0, terminal reward: +-1.0, probability: 0.1,0.8,0.1, S0:(0,0)

γ: 0.04 **γ: 0.06** **γ: 0.08** **γ: 0.09**



Analysis

If gamma is too low, policy iteration does not generate a valid solution for the large MDP. The action in the orange state is up. This is an effect of the negative terminal state. Without a large enough gamma, the impact of the negative terminal state cannot be overridden.

Value/Policy Iteration Summary

1. How many iterations does it take to converge?

	small MDP	**large MDP**
value iteration	15 iterations	36 iterations
policy iteration	5 iterations	19 iterations

2. Which one converges faster?

	small MDP	**large MDP**
value iteration	0 seconds	0.047 seconds
policy iteration	0 seconds	0.125 seconds

3. Although less policy iterations are required to converge, each iteration requires more calculation.
4. Policy and value iteration converge on the same answer if gamma is big enough and there are no equal decisions. If gamma is too small, policy iteration can converge on the wrong answer (experiment 9). If there is equal decision (neighbors of equal value) value iteration picks a random value where policy iteration picks the same value (based on experiment 8).
5. The number of states had no impact on value iteration with the exception of number of iterations and convergence time. Policy iteration failed to provide a valid policy with a large number of states and a small gamma.

Reinforcement Learning Algorithm - Q-Learning

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

"*According to this formula, a value assigned to a specific element of matrix Q, is equal to the sum of the corresponding value in matrix R and the learning parameter Gamma, multiplied by the maximum value of Q for all possible actions in the next state.*" [8]

```
Set the gamma parameter, and environment rewards in matrix R.
Initialize matrix Q to zero.
For each episode:
   Select a random initial state.
   Do While the goal state hasn't been reached.
      Select one among all possible actions for the current state.
      Using this possible action, consider going to the next state.
      Get maximum Q value for this next state based on all possible actions.
      Compute: Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
      Set the next state as the current state.
   End Do
End For
```

Figure Q-Learning algorithm

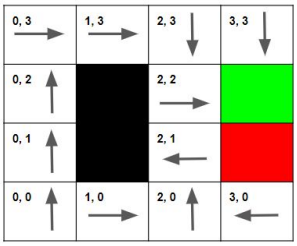
"*The algorithm above is used by the agent to learn from experience. Each episode is equivalent to one training session. In each training session, the agent explores the environment (represented by matrix R), receives the reward (if any) until it reaches the goal state. The purpose of the training is to enhance the 'brain' of our agent, represented by matrix Q. More training results in a more optimized matrix Q. In this case, if the matrix Q has been enhanced, instead of exploring around, and going back and forth to the same rooms, the agent will find the fastest route to the goal state.*" [8]

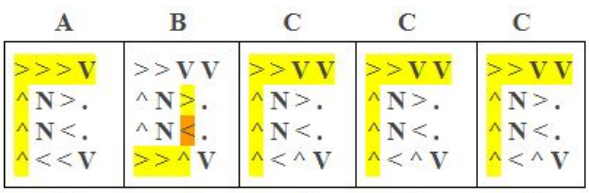
$$f(u,n) = \begin{cases} R^{+} \; if n < N_e \\ u \; otherwise \end{cases}$$

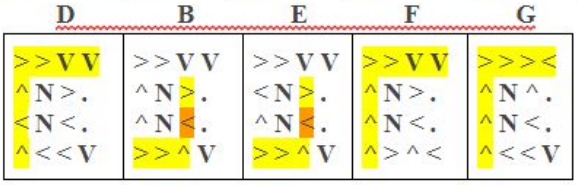
The exploration function determines how greed is traded off against curiosity. [9]

Experiment 10 - Solve small MDP using Q-Learning

reward: -0.04, terminal reward: +-1.0, probability: 0.1,0.8,0.1, S0:(0,0), γ: 0.9, Ne: 5, R+: 2







	State													
Action	(0, 1)	(3, 2)	(0, 0)	(1, 3)	(3, 3)	(3, 0)	(3, 1)	(2, 1)	(2, 0)	(2, 3)	(2, 2)	(1, 0)	(0, 2)	(0, 3)
(1, 0)	-0.17	0	-0.21	0.15	-0.04	-0.34	0	-0.94	-0.2	-0.04	0.29	-0.19	-0.15	0.08
(0, 1)	-0.02	0	-0.08	-0.1	-0.04	-0.92	0	-0.32	-0.15	-0.04	0	-0.21	0.03	-0.13
(-1, 0)	-0.17	0	-0.22	-0.1	0	-0.66	0	-0.1	-0.19	-0.08	-0.03	-0.2	-0.15	-0.13
(0, -1)	-0.17	0	-0.21	-0.11	0.3	-0.27	0	-0.56	-0.27	0.22	0	-0.21	-0.16	-0.11

Figure 16. Solving small MDP with Q-Learning (runtime 0.174 seconds)

Analysis

Due to the stochastic nature of Q-learning, the results change every time it is run. In 10 runs, 7 different policies were calculated. Run A, C, D, and F are valid. Runs B, E, and G were not valid. All the valid results were long routes. As with policy iteration, q-learning has issues with the negative terminal state.

Experiment 11 - Solve large MDP using Q-Learning

reward: -0.04, terminal reward: +-1.0, probability: 0.1,0.8,0.1, S0:(0,0), γ: 0.9, Ne: 5, R+: 2

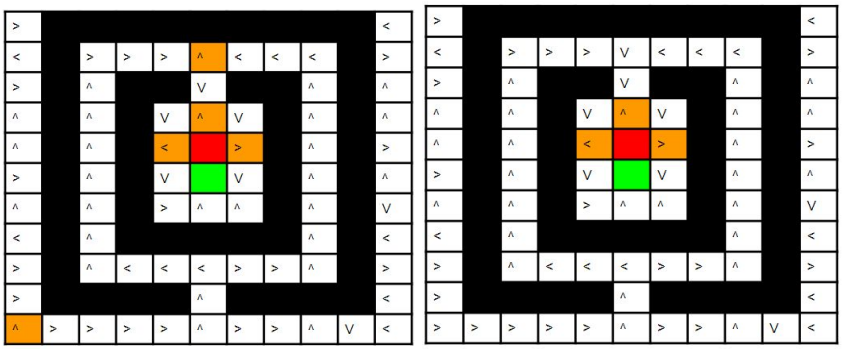
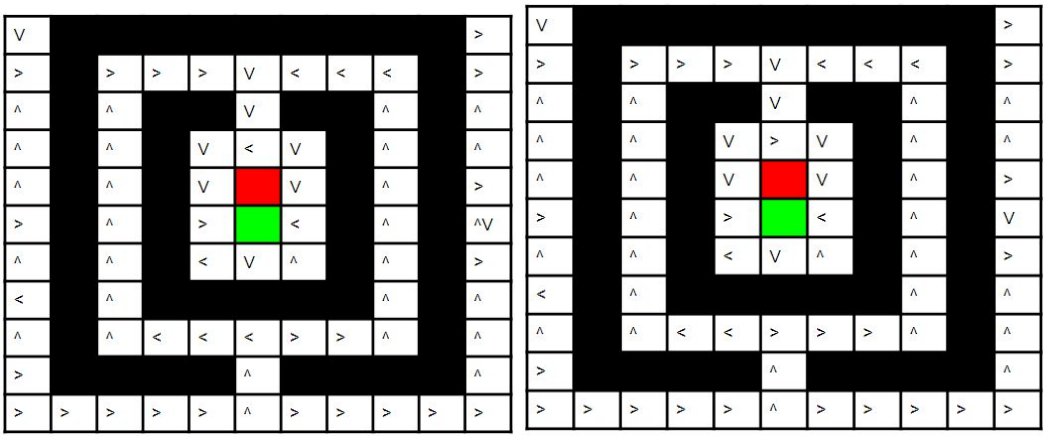


Figure 17. Orange boxes indicate state policies that do not lead to a solution

Analysis

Q-Learning did not converge when the reward was 0. As shown in figure 17, Q-learning does not result in a valid policy based on the parameters in this experiment.

Experiment 12 - Finding parameters to generate a valid Q-Learning policy



probability: 0,1.0,0 **negative terminal = 0**

Analysis

The failure to find a valid policy is caused by the negative terminal state. When the negative terminal state is set to 0 or the probability function is set to 0,1.0,0 the Q-Learning algorithm returns a valid policy.

Q-Learning Question/Answers

1) Q-Learning took longer (in real-time) than value or policy iteration. Q-learning did not converge in all the experiments (large MDP, reward=0).
2) I tried both greedy and exploration policy parameters (Ne, R+)
3) The exploration policy had no impact with such a large negative terminal function.

Summary

Both MDP's turned out ot be interesting. The small MDP highlighted aspects of value and policy iteration while the large MDP turned out to be difficult for the Q-learning algorithm. It was interesting how large of an impact the negative terminal state had on all the algorithms.

References

1. https://www.cs.cmu.edu/~katef/DeepRLControlCourse/lectures/lecture2_mdps.pdf
2. https://www.cs.cmu.edu/~katef/DeepRLControlCourse/lectures/lecture3_mdp_planning.pdf
3. Sutton and Barto, Reinforcement Learning: An Introduction, 1998
4. http://mas.cs.umass.edu/classes/cs683/lectures-2010/Lec13_MDP2-F2010-4up.pdf
5. https://lirias.kuleuven.be/bitstream/123456789/237416/1/thesis.pdf
6. http://www.cs.cmu.edu/~cga/ai-course/mdp.pdf
7. http://jmvidal.cse.sc.edu/library/watkins92a.pdf
8. http://mnemstudio.org/path-finding-q-learning-tutorial.htm
9. http://aima.cs.berkeley.edu/