Value Iteration and A*

Vasileios Papadopoulos

January 21, 2021

1 Environment

In a stochastic environment, for each action an agent performs there is a probability P(a) = 0.8 to succeed(to as planned) and 1-P(a) = 0.2 to move to different direction. For example: if the agent wants to move UP, there is a 0.8 probability to actually move UP and 0.2 probability to move either LEFT or RIGHT. If the agent wants to move RIGHT it will succeed with 0.8 probability and with 0.2 it will move LEFT or it will stay and the same state if it hits the wall. Figure below shows grid world and stochasticity. In case there is a wall in the direction that the agents wishes to move, then it stays put. The agent is in constant feedback loop with the environment meaning it takes an action α in state s and lands in a stochastic manner to state s' and gets a reward r.

2 Markov Decision Process (MDP)

In Markov decision process problems the goal is to find an optimal policy π* that gives the best action for each state. Optimal policy π* maximizes the expected sum of discounted(or not) rewards.

An MDP is defined by the following components:

1. Set of possible states: $S = \{s_0, s_1, ..., s_n\}$
2. Initial state: S_0
3. Set of possible actions: $A = \{a_1, a2, ..., a_m\}$
4. Transition model: $T(s, a, s')$
5. Reward function: $R(s)$

3 Value Iteration

Value iteration or Bellman update is a recursive dynamic programming algorithm. It is a method of computing the best values for each possible state an agent can be and eventually extracting the optimal policy π* for an MDP problem. The agent takes an action α from state s and it lands in state s' with a probability as this given by transition model .

$$V(s) \leftarrow \max_a \sum_{s'} T(s, a, s')[R(s, a, s') + V(s')] \quad (1)$$

Intuitively, the value of V(s) is the best action that maximizes the expected reward. Consider the grid world we defined before. In order to compute the value state of cell(3,3) we need to calculate all the possible future rewards for every possible action $A = \{a_1, a2, ..., a_m\}$, with discount factor $\gamma = 0.9$.

1. Agent tries to go right:
   - $V(\langle 3,3 \rangle)_{right} = \sum_{s'} T(\langle 3,3 \rangle, right, s')[R(\langle 3,3 \rangle), \gamma V(s')]$
   - $V(\langle 3,3 \rangle)_{right} = 0.9[0.8*1+0.1*0+0.1*0]$
   - $V(\langle 3,3 \rangle)_{right} = 0.72$
2. Agent tries to go left:
   - $V(\langle 3,3 \rangle)_{left} = \sum_{s'} T(\langle 3,3 \rangle, left, s')[R(\langle 3,3 \rangle), \gamma V(s')]$
   - $V(\langle 3,3 \rangle)_{left} = 0.9[0.8*0+0.1*0+0.1*0]$
   - $V(\langle 3,3 \rangle)_{left} = 0$
3. Agent tries to go down:
   - $V(\langle 3,3 \rangle)_{down} = \sum_{s'} T(\langle 3,3 \rangle, down, s')[R(\langle 3,3 \rangle), \gamma V(s')]$

- $V(\langle 3,3\rangle)_{down} = 0.9[0.8*0+0.1*0+0.1*1]$
- $V(\langle 3,3\rangle)_{down} = 0.09$

4. Agent tries to go up:
   - $V(\langle 3,3\rangle)_{up} = \sum_{s'} T(\langle 3,3\rangle, up, s')[R(\langle 3,3\rangle), \gamma V(s')]$
   - $V(\langle 3,3\rangle)_{up} = 0.9[0.8*0+0*0+0.1*1]$
   - $V(\langle 3,3\rangle)_{up} = 0.09$

Then we take the action that maximizes the value state.

- $V(\langle 3,3\rangle) = \max_a[V(\langle 3,3\rangle)_{right}, V(\langle 3,3\rangle)_{left}, V(\langle 3,3\rangle)_{down}, V(\langle 3,3\rangle)_{up}]$
- $V(\langle 3,3\rangle) = 0.72$

The above process is repeated for all states. After having calculated all values we repeat again until convergence which is guaranteed by Value iteration algorithm

3.1 Pseudo code

3.2 Discount factor

4 A* Algorithm

A* is a graph traversal and path search algorithm. It differs from Dijkstra algorithm as it uses best first search taking into account the current cost g and heuristic function h. It gives priority to nodes that are supposed to be better than others according to the value of function f(s) = g(s) + h(s) where h(s) is a heuristic function. For h(s) = 0, A* is similar to Dijkstra.

4.1 Pseudo code