# CS4710 - Artificial Intelligence: Homework 3

Fall 2016

Due: Wednesday, November 9, 2016

## Introduction

This time we will examine *value iteration*. In particular we are given the $7 \times 7$ world shown below.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | | |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | $\star$ |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |

## Reward Function

As you can see there is a special state indicated with the symbol $\star$ and with coordinates $(3, 6)$. The rationale is that at every time step, the reward of an agent at any state is $-1$, apart from the special $\star$ state where the reward is 0. In other words,

$$R(s) = \left\{ \begin{array}{rcl} 0 & , & \text{if } s = (3, 6) \\ -1 & , & \text{otherwise} \end{array} \right.$$

**Discount Factor.** We will be using discount factor $\gamma = 1$ throughout.

## Actions and Transition Model

At any given state, the agent can perform 9 actions, shown below.

$$\begin{array}{ccc} \nwarrow & \uparrow & \nearrow \\ \leftarrow & \cdot & \rightarrow \\ \swarrow & \downarrow & \searrow \end{array}$$

The eight of these actions are the traditional eight primary directions in the horizon ($N$, $NE$, $E$, $SE$, $S$, $SW$, $W$, $NW$) and as a result of each such move, the agent will move to a neighboring state in the respective direction. The agent also has a special move $stay$ (indicated with a $\cdot$ above) to its availability. Applying $stay$ at any state, does not alter the state to which the agent is.

**Respecting the Boundary.** The agent can never get off the grid. Hence, applying any action at any state *always leaves the agent within the grid.* For example, if the agent is at state $(0, 1)$ and applies $NE$ ($\nearrow$), then the resulting state is $(0, 2)$. That is, $\nearrow$ as well as $\rightarrow$ have the same next state $(0, 2)$ when applied to $(0, 1)$.

## Assignment Specifications, Writeup and Submission

You need to implement value iteration in three different cases as explained below. The different cases arise by the presence of wind; more information below.

In *every case* you need to

1. write down the value function that you computed (that is, all the entries of the $7 \times 7$ matrix), and then
2. write down the sequence of states and actions that are used when one follows the optimal policy $\pi^*$ starting from the state $(3, 0)$.

   Points to consider once you compute the value function and you want to present the sequence of states and actions due to $\pi^*$ are the following.

   (a) Omit the part of state(s) and action(s) that are repeated by following $\pi^*$ starting from $(3, 0)$, but clearly state which state(s) and action(s) are repeated.

   (b) If there is more than one such sequence generated by following $\pi^*$ when starting from $(3, 0)$, write down a second sequence of states and actions.

**Case I: No Wind.** In the first case, we apply value iteration in the world that is given in the introduction.

**Case II: Light Wind.** In the second case, wind blows along the columns with indices 3-5 from south to north. That is, when an action is performed for a state with a column in the range $\{3, 4, 5\}$, then the resulting row of the next state is reduced by one. For example, $(4, 4)$ is the next state of $(5, 4)$ when one applies *stay* in $(5, 4)$. Of course one has to respect the boundaries of our search space and thus, applying e.g. $\nearrow$ to state $(1, 5)$ leads to $(0, 6)$ (just like $\nearrow$ applied to $(1, 5)$ in the case of *no wind* would also lead to $(0, 6)$). Finally, applying $\searrow$ to state $(6, 5)$ leads to $(6, 6)$. In other words, *we enforce our constraints due to the boundary of our search space only in the end.*

**Case III: Strong Wind.** In the third and last case, the wind blows again from south to north along the columns with indices 3-5, but the effect is stronger. Hence, in this case, the row is reduced by 2. Again, we enforce our constraints due to the boundary of our search space only in the end of each move; e.g. applying $\searrow$ to $(6, 5)$ leads to $(5, 6)$.

**Submission.** Return one zipped file containing your code and your writeup.

## Algorithm for Value Iteration

The algorithm for value iteration is given in the book, but for convenience it is also given in Algorithm 1.

**Termination Condition.** Since we are using $\gamma = 1$ for this exercise, we need to modify the termination condition in Algorithm 1. We will use the following in line 10

$$\text{until } (\delta < \epsilon) \text{ or } (iters > MAX\_ITERS)$$

**Constants for the Execution.** We will use the following in every case:

$$\begin{cases} \gamma & = & 1 \\ \epsilon & = & 0.001 \\ MAX\_ITERS & = & 1000 \end{cases}$$

**Algorithm 1:** Value Iteration

```
Input: An MDP with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}(s)$, transition model $\mathbf{Pr}(s' \mid s, a)$, rewards $R(s)$,
       discount $\gamma$.
Output: The value (utility) of each state $s \in \mathcal{S}$.
1  Init($U'$);                          /* Initialize $U'$ (e.g. everything equal to 0) */
2  $iters \leftarrow 0$;
3  repeat
4      $iters \leftarrow iters + 1$;
5      $U \leftarrow U'$;
6      $\delta \leftarrow 0$;
7      foreach state $s \in \mathcal{S}$ do
8          $U'[s] \leftarrow R(s) + \gamma \cdot \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathbf{Pr}(s' \mid s, a) \cdot U[s']$;
9          if $|U'[s] - U[s]| > \delta$ then $\delta \leftarrow |U'[s] - U[s]|$ ;
10 until $\delta < \epsilon(1 - \gamma)/\gamma$;
11 return U';
```