

## Installation

We will be using the `conda` package management system.

To get started, download Anaconda from <https://docs.anaconda.com/anaconda/install/>.

After downloading the code for the homework, `cd` to the directory of the code.

A list of all the packages needed for the virtual environment is in `spec-file.txt`. After downloading Anaconda, duplicate the virtual environment with:

```
conda create --name <your_environment_name> --file spec-file.txt
```

Then, activate your environment

```
conda activate <your_environment_name>
```

When you are finished using the environment you can deactivate it with:

```
conda deactivate
```

You can always activate it again with

```
conda activate <your_environment_name>
```

## Running the Code

You can run the code with the command `python run_assignment_1.py --map maps/map1.txt`. You should see a visualization of the AI Farm environment. The code will output the  $\Delta$  from Algorithm 2 at every step and output DONE when value iteration has converged.

There are switches that you can use

- `--discount`, to change the discount (default=1.0)
- `--no_text`, to omit text shown on the figure
- `--no_val`, to omit the state-value shown on the figure
- `--no_policy`, to omit the policy shown on the figure
- `--rand_right`, to change the probability that the wind blows you to the right (default=0.0)
- `--wait`, the number of seconds to wait after every iteration so that you can visualize your algorithm (default=0.0)

i.e. `python run_assignment_1.py --map maps/map1.txt --no_text --rand_right 0.1 --wait 0.5`

# Helper Functions

In your implementations, you are given an `Environment` object `env`. You will need the `env.get_actions()` function that returns a list of all possible actions. You will also need `env.state_action_dynamics(state, action)`, which returns, in this order, the expected return  $r(s, a)$ , all possible next states given the current state and action, their probabilities.

Keep in mind, while the notation for value iteration update,  $V(s) \leftarrow \max_a(r(s, a) + \gamma \sum_{s'} p(s'|s, a)V(s'))$ , sums over all possible states, `env.state_action_dynamics(state, action)` omits all states that have a state-transition probability of zero. This significantly reduces the number of elements in the summation.

## Part 1: Value Iteration and the Optimal Policy

### Part 1.1: Value Iteration

Value iteration can be used to find, or approximate, the optimal value function. In this exercise, we will be finding the optimal value function, exactly.

Implement `value_iteration_step` in `assignments_code/assignment1.py`.

Upon running `python run_assignment_1.py --map maps/map1.txt`, Algorithm 1 will automatically begin running with the value  $\theta$  set to 0. It will automatically call your code, Algorithm 2.

Vary the environment dynamics by setting `--rand_right` to 0.1 and 0.5. The resulting state-value function and policy should be the same as shown in the slides on Markov Decision Processes.

---

#### Algorithm 1 Value Iteration

---

```
1: procedure VALUE_ITERATION( $\mathcal{S}, V, \theta, \gamma$ )
2:    $\Delta \leftarrow \text{inf}$ 
3:   while  $\Delta > \theta$  do
4:      $\Delta, V = \text{Value\_Iteration\_Step}(\mathcal{S}, V)$ 
5:   end while
6:   return  $V$ 
7: end procedure
```

▷ Approximation of  $v_*$

---

---

#### Algorithm 2 Value Iteration Step

---

```
1: procedure VALUE_ITERATION_STEP( $\mathcal{S}, V, a\gamma$ )
2:    $\Delta \leftarrow 0$ 
3:   for  $s \in \mathcal{S}$  do
4:      $v \leftarrow V(s)$ 
5:      $V(s) \leftarrow \max_a(r(s, a) + \gamma \sum_{s'} p(s'|s, a)V(s'))$ 
6:      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
7:   end for
8:   return  $\Delta, V$ 
9: end procedure
```

---

## Part 1.1: Obtaining the Optimal Policy

After finding the optimal value function  $v_*$ , we can then use it to find the optimal policy using 1. Implement `get_action` in `assignments_code/assignment1.py` so that it implements 1.

$$\pi^*(s) = \operatorname{argmax}_a (r(s, a) + \gamma \sum_{s'} p(s'|s, a) v_*(s')) \quad (1)$$

### Part 1: What to Turn In

Turn in your implementation of `assignments_code/assignment1.py`.

## Part 2: Comparison to Sutton and Barto's Notation

You will notice that the state-value assignment step in Algorithm 2 is written differently than in the value iteration algorithm on page 83 of *Reinforcement Learning: An Introduction* (<http://incompleteideas.net/book/RLbook2020.pdf>):

$$V(s) \leftarrow \max_a (r(s, a) + \gamma \sum_{s'} p(s'|s, a) V(s')) \quad (2)$$

$$V(s) \leftarrow \max_a \sum_{s', r} p(s', r|s, a) (r + \gamma V(s')) \quad (3)$$

Show that these two are equivalent by rearranging 3 so that it looks like 2. Use the definitions provided in the lecture slides on Markov Decision Processes. Show your work.

### Part 2: What to Turn In

Turn in a PDF of your work.