CSE 471 Fall 2023 Final Exam Project

Copyright © 2023 Arizona Board of Regents

Topics

- Introduction
- Exam Policy
- The Environment
- Files provided
- Input Parameters
- Overview of Questions
- Evaluation
- Demo
- Questions

Copyright © 2023 Arizona Board of Regents

Introduction

Autonomous Driving Domain

- Given: Discretized grid-based simulation of highway scenario
- To Do: Encode different agent functions for the autonomous driving agent ‘A’

Arizona State University

Copyright © 2023 Arizona Board of Regents

Exam Policy

- Independent work
  - Collaboration or discussion is NOT allowed
- All materials in the project are copyrighted
  - Copying or transmitting code in any form is NOT allowed
- Rules for Ed discussion
  - Only post clarification questions
  - No questions on solving the problems

Arizona State University

Copyright © 2023 Arizona Board of Regents

The Environment

Autonomous Driving Domain

- Discretized grid representation of road
- Random initialization of cars
- Autonomous Agent ‘A’ can navigate by:
    - moving forward ‘F’
    - moving left ‘L’
    - moving right ‘R’
    - waiting ‘W’
- All cars navigate from the bottom to top of the grid

Arizona State University

Copyright © 2023 Arizona Board of Regents

Files Provided

Autonomous Driving Domain

driving.py	main file to run
environment.py	simulator code
state.py	your interface to access the simulator
agent.py	your code goes here
problem.py	initializes agent controller
autograder.py	grades your project

Copyright © 2023 Arizona Board of Regents

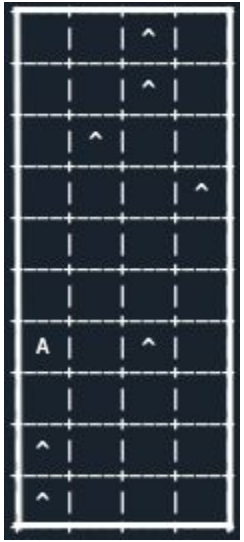
Input Parameters

Autonomous Driving Domain

--agent	specify the agent type
--height	specify length of the road (number of rows)
--width	specify the number of lanes in the road (number of columns)
--occupancy	specify the % of cells to be populated with other cars (0 - 0.3)
--init	specify the initial location of ‘A’ (0 - 0.9)
--seed	Random initialization of other cars

occupancy = 0.2
init = 0.2

height = 10



width = 4

Copyright © 2023 Arizona Board of Regents

Overview of Questions

Autonomous Driving Domain

- Q1) Simple Reflex Agent (5 points)
- Q2) Expectimax Agent (10 points)
- Q3) Learning Agent (10 points)

Arizona State University

Copyright © 2023 Arizona Board of Regents

Q1) Simple Reflex Agent

Autonomous Driving Domain

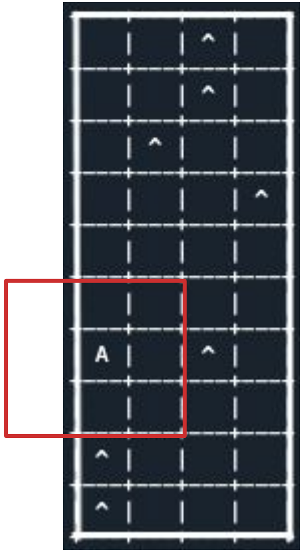
Given: Localized view of the road

-1	0	0
-1	2	0
-1	0	0

0	0	0
0	2	0
0	0	0

1	1	-1
0	2	-1
0	0	-1

10	10	10
0	2	-1
0	0	-1



Copyright © 2023 Arizona Board of Regents

Q1) Simple Reflex Agent

agent.py

```
class ReflexAgent(Agent):
    """
    A reflex agent chooses an action at each choice point by following simple rules

    """

    def get_action(self, percept):
        """
        Description
        -----------
        This function returns the reflex action given the current percept.
        The percept is essentially a (3 x 3) grid sized partial view of the road environment,
        with 'A' at the center i.e. at index (1, 1).

        """
        "*** YOUR CODE HERE ***"

        raise Exception("Function not defined")
```

Arizona State University

Copyright © 2023 Arizona Board of Regents

Q2) Expectimax Agent

Given: Full view of the road via ExpectimaxState in state.py

def get_legal_actions(self, car_index):	def get_min_distance_to_goal(self, AA_loc):
def generate_successor(self, car_index, action):	def get_score(self):
def apply_AA_action(self, action):	def is_car_on_road(self, car_index):
def apply_action(self, car_index, action):	def is_done(self):
def get_num_cars(self):	def is_crash(self):
def get_car_position(self, car_index):	

Arizona State University

Copyright © 2023 Arizona Board of Regents

Q2) Expectimax Agent

agent.py

```
class ExpectimaxAgent(Agent):
    """
    An expectimax agent chooses an action at each choice point based on the expectimax algorithm.
    The choice is dependent on the self.evaluationFunction.

    All other cars should be modeled as choosing uniformly at random from their legal actions.

    """
    def __init__(self, depth=3):
        self.index = 0 # 'A' is always agent index 0
        self.depth = int(depth)
        super().__init__()

    def evaluation_function(self, road_state):
        """
        Description
        -----------
        This function returns a score (float) given a state of the road.

        """
        "*** YOUR CODE HERE ***"

        raise Exception("Function not defined")

    def get_action(self, road_state):
        """
        Description
        -----------
        This function returns the expectimax action using self.depth and self.evaluationFunction.
        All other cars should be modeled as choosing uniformly at random from their
        legal moves.

        """
        "*** YOUR CODE HERE ***"

        raise Exception("Function not defined")
```

Copyright © 2023 Arizona Board of Regents

Q3) Learning Agent

Approximate Q Learning Agent

--episodes	specify number of episodes for training
--features	specify the number of features defined

Copyright © 2023 Arizona Board of Regents

Q3) Learning Agent

Given: Full view of the road via LearningState in state.py

def is_terminal(self):
def step(self, action):
def get_road_data(self):
def get_car_locations(self):
def get_height(self):
def get_width(self):

Copyright © 2023 Arizona Board of Regents

Q3) Learning Agent

agent.py

```python
class LearningAgent(Agent):
    """
    A learning agent chooses an action at each choice point based on the Q values approximated.
    In this project your learning agent is essentiually an ApproximateQLearningAgent

    """

    def __init__(self, num_features=4, custom_weights=False, weights=None, alpha=0.1, gamma=0.99):
        self.alpha = alpha
        self.gamma = gamma
        self.num_features = num_features
        self.decay_rate = 0.99
        self.epsilon = 1

        if custom_weights:
            self.weights = weights
        else:
            self.weights = np.random.rand(num_features)

        super().__init__()

    def get_weights(self):
        return self.weights

    def get_features(self, state, action):
        """
        Description
        ----------
        This function returns a vector of features for the given state action pair

        Compute: f_1(s, a), f_2(s, a), ... , f_n(s, a)

        """
        "*** YOUR CODE HERE ***"

        raise Exception("Function not defined")

    def get_Q_value(self, state, action):
        """
        Description
        ----------
        This function returns the Q value; Q(state,action) = w . featureVector
        where . is the dotProduct operator

        Compute: Q(s, a) = w_1 * f_1(s, a) + w_2 * f_2(s, a) + ... + w_n * f_n(s, a)

        """
        "*** YOUR CODE HERE ***"

        raise Exception("Function not defined")
```

```python
    def compute_max_Q_value(self, state):
        """
        Description
        ----------
        This function returns the max over all Q(state, action)
        for all legal/available actions for the given state
        Note that if there are no legal actions, which is the case at the
        terminal state, you should return a value of 0.0.

        Compute: max_a' Q(s', a')

        """
        "*** YOUR CODE HERE ***"

        raise Exception("Function not defined")

    def update(self, state, action, next_state, reward):
        """
        Description
        ----------
        This function updates the weights based on the given transition

        """
        "*** YOUR CODE HERE ***"

        raise Exception("Function not defined")

    def get_action(self, state):

    def train(self, state):

def normalize(value, min_value, max_value):
    """
    Description
    ----------
    Normalizes a given value between 0-1

    """
    return (value - min_value) / (max_value - min_value)

def manhattan_distance(loc1, loc2):
    """
    Description
    ----------
    Returns the Manhattan distance between points loc1 and loc2

    """
    return abs( loc1[0] - loc2[0] ) + abs( loc1[1] - loc2[1] )
```

Copyright © 2023 Arizona Board of Regents

Evaluation

autograder.py

--q	specify the question to grade
--features	specify the number of features defined
--verbose	Specify True if you want to see detailed output

Grading:

- Passing 70% of test cases for a given questions is awarded full points

Copyright © 2023 Arizona Board of Regents