Intelligent systems 2015-2016

Seminar Assignment 2

The assignment is to be done in **pairs**. The presentations will be held during the lab practices in the week of **January 18 – 22.**

The topic of the assignment is reinforcement Learning. The task is to learn a policy for driving on a simulated multilane highway. Your agent (vehicle) should stay on the road avoiding collisions with other vehicles. The agent has 5 actions available:

Action	Description
1	drive forward (the velocity is unchanged)
2	steer to the left (the velocity is unchanged)
3	steer to the right (the velocity is unchanged)
4	increase the velocity (drive forward)
5	increase the velocity (drive forward)

The agent's goal is to drive as far as possible, i.e. indefinitely. The drive ends if the agent hits the roadside curb or other vehicle, or if it runs out of fuel.

On the course web page you can find two files, "**simulation.R**" and "**RL.R**". The first file contains functions for learning and running the traffic simulations. Please do not change any of these functions.

The second file contains the main program that initializes the variables, carries out the learning task, and starts the simulation using the learned policy. The file also contains two functions that represent the main focus of the assignment: **getStateDesc** and **getReward**. Your task is to complete the implementation of these functions to enable efficient learning.

The getStateDesc function

The ***getStateDesc*** function is automatically called whenever there is a change in the simulated environment. The function receives the information about the objects on the road that are inside the agent's field of view (the ***sceneObjects*** argument). The function returns the description of the discrete state, which represents a concise description of the situation on the road.

```
getStateDesc <- function(sceneObjects) {
    …

    state
}
```

The ***sceneObjects*** argument is a two-dimensional table (data.frame). It contains the following columns: *type*, *xtopleft*, *ytopleft*, *xbottomright*, *ybottomright*, *speed,* and *fuel*. The first column describes the type of an object and can take one of the following values:

Object type	Description
mycar	the agent's car
car	other vehicles in traffic
fuel	additional fuel packages
leftside	the left roadside curb
rightside	the right roadside curb

The values listed from the second to the fourth column refer to the position of an object, relative to the agent's position (see Figure 1). The vertex naming convention is shown in Figure 2.

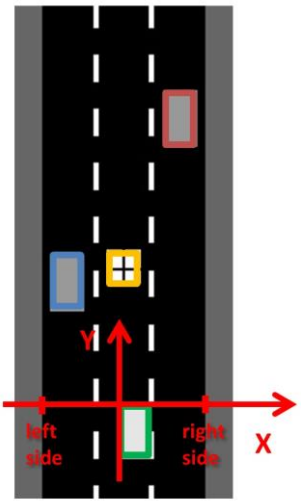

Figure 1: The origin of the coordinate system is set at the top left point of the agent's car.

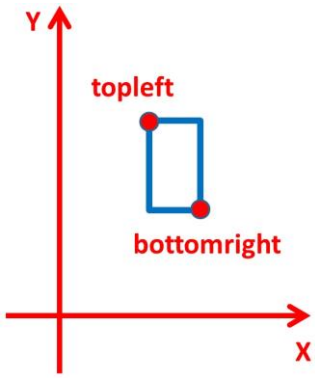

Figure 2: Vertex naming convention.

For the situation shown in Figure 1 the sceneObjects table would contain the following rows:

	type	xtopleft	ytopleft	xbottomright	ybottomright	speed	fuel
1	mycar	0	0	12	-20	5	4530
2	leftside	-28	NA	-28	NA	NA	NA
3	rightside	35	NA	35	NA	NA	NA
4	fuel	-3	61	9	49	0	NA
5	car	18	117	30	97	4	NA
6	car	24	57	-12	37	4	NA

The first row refers to the agent (shown in Figure 1 as the white rectangle with the green border). The second and the third row present the position of the left and the right roadside curb, respectively. The fourth row relates to the fuel package (shown in Figure 1 as the white rectangle with the yellow border). The last two lines describe the position and speed of other vehicles on the road (shown in Figure 1 as the rectangles with the brown and blue borders).

The purpose of the ***getStateDesc*** function is to summarize all relevant information about the situation on the road using the provided list of objects in the agent's surroundings. A simple initial idea is to describe the situation with the respect to the presence of other vehicles in the immediate vicinity of the agent's car. For example, one can divide the space around the agent's car into three areas (left, front, right). In this case, the current situation on the road can be represented using a three dimensional vector. Each element describes the presence of other objects inside the particular area (value 1 means there is no object in the area, value 2 means that there is an object present in the area). Using this encoding scheme, the situations shown in Figure 3 and Figure 4 would be described with vectors c(1, 1, 1) and c (1, 2, 1), respectively.

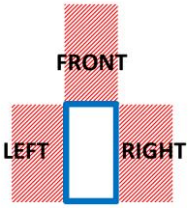

Figure 3: There are no other vehicles in the areas around the agent's car.

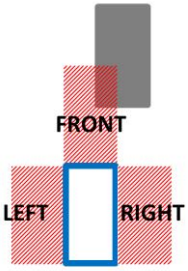

Figure 4: There is another vehicle in the front of agent's car.

This state description is still incomplete since it does not include other important information like the distance to the curbs on either side of the road, the position of fuel packages, etc.

Your task is to expand (or completely redesign) the state encoding scheme and implement it in the *getStateDesc* function. The state space must be finite and discrete, so you should consider the following limitations:

- **the resulting vectors returned by the *getStateDesc* function must be of the same length,**
- **all elements in the resulting vectors must be positive integers.**

In order to improve the speed and reliability of learning, it is desirable to keep the state space as small as possible!

The getReward function

The ***getReward*** function is automatically called during the learning process. The function gets a description of the current state (the state argument), the last action performed by the agent (the action argument) and the list of collided objects (the hitObjects argument, a list containing the types of objects that the agent collided with during the last action; if the agent did not collide with any other object, the list is empty). The result of the function is the reward (or punishment) that the agent collects after performing the specified action.

```
getReward <- function(state, action, hitObjects) {
    …

    reward
}
```

Your task is to implement meaningful rewards in the getReward function. The returned reward should encourage the agent to perform useful manoeuvres (such as collecting additional fuel packages) and to discourage harmful manoeuvres (such as colliding with other vehicles or the roadside curbs).

Q-learning

After you have implemented the ***getStateDesc*** and ***getReward*** functions, you may proceed to the learning process. The ***qlearning*** function has been implemented in the "**simulation.R**" file. The function requires the following arguments:

```
qlearning(dimStateSpace, gamma = 0.9, maxtrials = 200, maxdistance = 100000)
```

The ***dimStateSpace*** argument is a vector of the same length as the state description returned by the ***getStateDesc*** function. Individual elements of the ***dimStateSpace*** vector define the maximum values of the corresponding elements that can be returned by the ***getStateDesc*** function (in other words, the ***dimStateSpace*** vector defines the size of the state space).

The rest of the arguments in the ***qlearning*** function are optional and used to determine the discount factor (the gamma argument), the maximum number of learning episodes (the maxtrials argument) and the longest distance travelled in one episode (the maxdistance argument).

The ***qlearning*** function returns a matrix needed to carry out simulations.

The main program

The main program starts with the initialization of constants. The ***initConsts*** function takes two arguments. The first argument specifies the number of road lanes (an integer between 2 and 6), while the second argument determines the average number of vehicles ahead of the agent's car (an integer between 1 and 10). After the initialization, the agent's learning is performed. Finally, the simulation can be started using the returned matrix.

```
initConsts(numlanes=3, numcars=1)
qmat <- qlearning(dimStateSpace = c(2, 2, 2))
simulation(qmat)
```

Specific tasks

- implement the ***getStateDesc*** and ***getReward*** functions,
- train the agent to drive through the simulated environment,

- analyse the results (the average distance travelled) as a function of the given parameters (the state description, the reward function, the number of road lanes, the number of vehicles on the road, the discount factor, etc.),
- present the findings in the report (a document in pdf or doc format).

Your point score will be based on your exposition and justification of the chosen approach, the achieved results and your interpretation of the results.