COMP3211

Homework 1

LAM, Pak Ho

A COMP3211 Written Assignment

February 13, 2026

Question 1

1.1. No, we have the 4 sensors of s_2, s_4, s_6, s_8, and such agent can be designed as

$$\begin{aligned} s_2\bar{s_4} &\rightarrow \text{east} \\ s_4\bar{s_6} &\rightarrow \text{south} \\ s_6\bar{s_8} &\rightarrow \text{west} \\ s_8\bar{s_2} &\rightarrow \text{north} \\ 1 &\rightarrow \text{north} \end{aligned}$$

When there is a wall above the robot and the east side is cleared, we will move east. When there is a wall in the east side and the south side is cleared, we will move south. When there is a wall in the south side and the west side is cleared, we will move west. When there is a wall in the west side and the north side is cleared, we will move north. And if theres no wall around we are default to move the robot to north. However, this rule set fails because there exist position where the local pattern s_2, s_4, s_6, s_8 is equal but the actual action should be different. Consider a case such that the robot walk around the inner obstacle, when it walk through the corners all sensors will return 0. Since in the actual case only s_1, s_3, s_5 or s_7 is 1, in the local sensors, all these position is returned as all 0. A memoryless agent seeing (0,0,0,0) at all corners must pick the same action (north), which works at one corner but fails at the other three. So all positon will lead to the action North, however, if we want to make the robot follow the rectangle outer boundary located in the middle of the map, they should perform east, west, south or north respectively, so the local sensors cannot distinguish these positions.

1.2 Sometimes we can or cannot "compute" s_7 out. Normally, with memory we can set $w_7 = 1$ iff in the previous step $w_8 = 1$ and the robot move north previously. We have the design

$$\begin{aligned} s_2\bar{s_4} &\rightarrow \text{east} \\ s_4\bar{s_6} &\rightarrow \text{south} \\ s_6\bar{s_8} &\rightarrow \text{west} \\ s_8\bar{s_2} &\rightarrow \text{north} \\ w_3 &\rightarrow \text{east} \\ w_5 &\rightarrow \text{south} \\ w_7 &\rightarrow \text{west} \\ w_1 &\rightarrow \text{north} \\ 1 &\rightarrow \text{north} \end{aligned}$$

With:

- $w_1 = 1$ iff at the previous time step, $w_2 = 1$, and the robot moved east;

- $w_3 = 1$ iff at the previous time step, $w_4 = 1$, and the robot moved south;
- $w_5 = 1$ iff at the previous time step, $w_6 = 1$, and the robot moved west;
- $w_7 = 1$ iff at the previous time step, $w_8 = 1$, and the robot moved north.

However, notice that this robot moves in clockwise way in outer boundary, such that when we approach the southeast corner, such that s_5 is true. But, in the actual case if the outer boundary is a simple rectangle, s_3, s_5 and s_7 are all 1, which is different from what we predict. The w_7 we are predicting are only used for tracking the outer boundary, so that when we reach the southwest corner we will go north, which does not computer s_7 in any situations.

However for some occasions we can compute s_7 out. For example, when previous step is recored as moving north, and w_8 is recored as 1 previously, which means we are moving one step north now, we can predict and compute out that $s_7 = 1$ since we have passed it previously.

Question 2

2.1. Consider all possible cases: We notices that when x_3 is 1 as long as x_2 is 0 then the TLU will reach a value of 3 or above regardless of other input, which give us the first term $x_3 \wedge \bar{x_2}$ and when x_3 is 1 or any of x_1 or x_4 is 1, then it will also be more than 3 regardless of values of x_2 and x_5 (even both of them are 1) the value computed is more than 3. Which give us the second term $x_3 \wedge (x_1 \vee x_4)$. Lastly, when only x_1 and x_4 is 1, x_5 is acceptable to be 1 since its value is larger than 3 but x_2 must not be 1 as it will lower the value to lower than 3, so the third term is $(x_1 \wedge x_4) \wedge \bar{x_2}$ Combining the term, the boolean function is:

$$f = \bar{x_2} \wedge (x_3 \vee (x_1 \wedge x_4)) \vee (x_3 \wedge (x_1 \vee x_4))$$

2.2 The question ask for boolean function which threshold is larger or equal to 10. However, when we evaluate the maximum of the TLU, which we assign value 1 to all positive values and assign value 0 to all negative values, we compute: $1.9 \times 1 - 2.5 \times 0 + 5.2 \times 1 + 2.3 \times 1 - 0.5 \times 0 = 9.4 < 10$. As the max value is also small than 10, any inputs of this TLU will output a boolean result of false, we can directly state that the boolean function is

$$f = \text{false}$$

Question 3

3.1 My fitness function is designed as accuracy of the prediction, calculated by

$$\frac{\#\text{numbers of correct predictions}}{\#\text{total number of predictions performed}}$$

3.2 My crossover function will iterate through every weights of the program, and randomly select it from one of its parents to form a new program. If the program predict correctly it will be awarded 1 point and 0 otherwise. Finally, np mean function is used to find its accuracy.

3.3 My copy operator is runned by breaking the programs up to 10 groups from all programs, those groups are sent into tournament selection, and the best 10 programs, which are winners, will be selected

3.4 The mutator function usage is kept minimal and applies it to only 1 equation, a randomly selected weight of the program will be randomised

3.5 1000 programs are initialised initially, all programs' prarmeters are assigned values between -1 to 1 in float numbers, generated uniformly, this is achieved by numpy function.

3.6 The stopage of evolution is determined by a fixed iterator, this iterator is fine tuned few times during local test and finally it is fixed in 100 epoches.

3.7 The output is as follows:

```
Prediction of parameters:  [-0.30942699 -0.25004898 -0.92231307  0.65589858
    0.48219909 -0.47080655 0.66577782 -0.24509463  0.14474253]
Accuracy:  0.95
```

This set achieved a 95% accuracy against the training set.