COMP5211 Advanced Artificial Intelligence
Assignment 1

Zheng Yunchuan
UST ID: yzhengbj

October 5, 2020

Problem 1

Let me adopt the notation in the lecture slides. Then a production system that can control the robot to reach bottom-left corner is as following.

$$\begin{aligned} \overline{s_8} &\rightarrow west, \\ \overline{s_6} &\rightarrow south, \\ 1 &\rightarrow nil. \end{aligned} \tag{1}$$

Problem 2

Let us denote the five inputs X_1, X_2, X_3, X_4, X_5, where $X_i \in \{0, 1\}$.

Notice that the coefficient of X_5 is 0.5. With simple calculation, it is easy to know that the value of X_5 does not affect the excess of threshold. Therefore X_5 is not necessary to be in the function.

According to the weight vector, only when X_1 or X_2 is true could the threshold be passed. If X_1 is true and X_2 not, then the threshold will be passed as long as neither X_3 nor X_4 is true. This case can be expressed as $X_1 \wedge \neg X_2 \wedge \neg X_3 \wedge \neg X_4$.

When X_2 is true and X_1 not, the threshold is passed as long as not both X_3 and X_4 are true, i.e. $\neg X_1 \wedge X_2 \wedge \neg(X_3 \wedge X_4)$.

Finally, if both X_1 and X_2 are true, the threshold will always be passes, i.e. $X_1 \wedge X_2$.

In summary, we get the boolean function for the TLU:

$$(X_1 \wedge \neg X_2 \wedge \neg X_3 \wedge \neg X_4) \vee (\neg X_1 \wedge X_2 \wedge \neg(X_3 \wedge X_4)) \vee (X_1 \wedge X_2) \tag{2}$$

Problem 3

1. My fitness function is the number of correct predictions. First let me define a threshold function

$$g(x, \theta) = \begin{cases} 1, & x \geq \theta \\ 0, & otherwise \end{cases} \tag{3}$$

   The fitness function is

$$f(w, \theta) = \sum_{i=1}^{m} |l_i - g(w^T x_i, \theta)| \tag{4}$$

2. My crossover operator for 90% of generation $n+1$: a mother and a father is chosen from generation n by the tournament selection process. Then a random position of tuples (except the labels) is pick, which divides the mother and the father tuple into two parts. The father's front part is concatenated with the mother's back part, and vice versa.
3. My copy operator for 10% of generation $n+1$: Copy the top 10% from generation n.
4. I use mutate operators on 1% of each generation. For each of the picked, from a random position, replace the back parts with uniformly distributed weights. Meanwhile, I replace their thresholds with uniformly distributed numbers.
5. The size of the initial generation is set to 5000, and programs generated by randomly generating the uniformly distributed weights and thresholds.
6. The condition for termination is finishing 100 iterations, or getting all predictions right on the training set.
7. The output of my system during one execution is [-0.69959502, -0.87233665, -1.87410389, 0.96406645, 1.09844068, -0.78799784, 1.93690851, -0.4921522, 0.72416626, 0.11763949].

Problem 4

1

The weight vector for going north in my learning is $[0.1, -0.2, -0.2, 0.0, 0.0, 0.0, 0.0, 0.1]$, where the last element in the vector corresponds to the special input 1. Therefore the boolean expression that corresponds to my perceptron for the move north action is:

$$s_1\overline{s_2 s_3}s_8. \tag{5}$$

2

Implemented in class `ECAgent` in `reactiveAgent.py`.

Problem 5

1

Implemented in class `SMAgent` in `reactiveAgent.py`.

2

No, we can't. Because in this case we have previous sensory input and previous action in the feature vector. Those features are not independent, so we cannot capture the training set by a linearly separable Boolean function. Therefore, the Error-Correction procedure does not work here.