CSIT6000F Assignment #1 Suggested Solution

Problem 1. (30%)

(a)

A simple solution is as follows:

$$\begin{aligned}\bar{s_2} &\rightarrow North\\ \bar{s_8} &\rightarrow West\\ 1 &\rightarrow Nil\end{aligned}$$

You can also use the solution on the lecture note:

$$\begin{aligned}inCorner &\rightarrow Nil\\ 1 &\rightarrow bf\end{aligned}$$

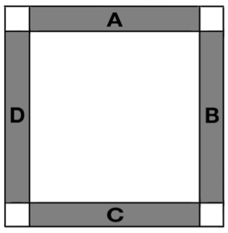
where bf is the boundary following production system, and define $inCorner$ as:

$$inCorner \text{ iff } s_2 s_3 s_4 + s_4 s_5 s_6 + s_6 s_7 s_8 + s_8 s_1 s_2$$

(b)

We divide the grid into two regions: the inner 8×8 grid, and the others (the outer region).

- If the initial position is in the inner 8×8 grid, then only one action is to be taken until the robot hits the boundary as inputs for the eight sensors of each cell in the inner 8×8 grid are all the same (all zero). w.l.o.g. we assume that single action is moving North. As a result, the robot will end up in region A, as illustrated below.

  Since the perceptions for cells in region A,B,C and D, respectively, are all the same, only one action is allowed for each region. If the four actions for A, B, C and D are all not going back to the region, then the robot will stay in the outer region forever, hence cannot visit every cell. If any of these four actions is stepping into the inner grid, the robot will end up in a loop:

  {goes into the inner region $\rightarrow$ go North till region A $\rightarrow$ goes into the inner region from exactly the same position as last time}.

  Therefore it cannot visit every cell neither.

- If the initial position is in the outer region, the proof is the same as the latter part above. The robot will either stay in the outer region forever or go in and out in a loop.

(c) Using method 1 in part (a), we can rst let the robot go to the left upper corner of the grid. Once the robot is in the left upper corner, we then let it traverse the cells one by one.

Two states are needed:

PA denotes the previous action, and **N** for North, **S** for South, **W** for West and **E** for East.

Corner indicates whether the robot has been to the left upper corner. **1** for yes, **0** for no.

Initially, **PA** = Null and **Corner** = 0.

The production system is as follows:

$$\overline{\textbf{Corner}}\ \bar{s_2} \rightarrow North \tag{1}$$
$$\overline{\textbf{Corner}}\bar{s_8} \rightarrow West \tag{2}$$
$$\overline{\textbf{Corner}} \rightarrow Set\ \textbf{Corner} = 1\ \textbf{and PA} = \textbf{E} \tag{3}$$
$$(\textbf{PA} = \textbf{E})s_2 \rightarrow South \tag{4}$$
$$(\textbf{PA} = \textbf{S})\bar{s_6} \rightarrow South \tag{5}$$
$$(\textbf{PA} = \textbf{S}) \rightarrow East \tag{6}$$
$$(\textbf{PA} = \textbf{E}) \rightarrow North \tag{7}$$
$$(\textbf{PA} = \textbf{N})\bar{s_2} \rightarrow North \tag{8}$$
$$(\textbf{PA} = \textbf{N}) \rightarrow East \tag{9}$$
$$s_2 s_4 \rightarrow Nil \tag{10}$$
$$\tag{11}$$

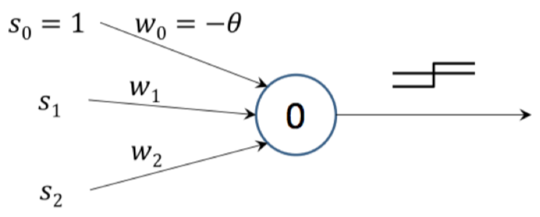
Problem 2. (15%)

$$x_1x_2 + x_1\bar{x_3}\bar{x_4} + x_2(\bar{x_3} + \bar{x_4})$$

Problem 3. (20%)

The maximal set of training instances for a logical disjunction with 2 inputs is shown below:

s_1	s_2	d
1	1	1
1	0	1
0	1	1
0	0	0



A mininal training set could be:

$$\{(s_0 = 1, s_1 = 1, s_2 = 1, d = 1), (s_0 = 1, s_1 = 0, s_2 = 0, d = 0)\}$$

	$W_{old} = (w_0, w_1, w_2)$	$X = (x_0, x_1, x_2)$	d	$\sum$	f	$d = f?$	$W_{new} = (w_0, w_1, w_2)$
1	$(0, 0, 0)$	$(1, 1, 1)$	1	0	1	*yes*	$(0, 0, 0)$
2	$(0, 0, 0)$	$(1, 0, 0)$	0	0	1	*no*	$(-1, 0, 0)$
3	$(-1, 0, 0)$	$(1, 1, 1)$	1	-1	0	*no*	$(0, 1, 1)$
4	$(0, 1, 1)$	$(1, 0, 0)$	0	0	1	*no*	$(-1, 1, 1)$
5	$(-1, 1, 1)$	$(1, 1, 1)$	1	1	1	*yes*	$(1, 1, 1)$
6	$(-1, 1, 1)$	$(1, 0, 0)$	-1	0	0	*yes*	$(1, 1, 1)$

After interation 5 & 6, we got the converging sequence of weights $(-1, 1, 1)$, and this weight vector works correctly with all the training instances in the maximal set. Therefore, $(1, 1)$ and $(0, 0)$ are the minimal set.

Problem 4. (35%) There are no unique solution to this problem. Below is a sample:

1. What's your fitness function?

   For each instance in the training set, if the given program agrees with the label, then it gets 0 point, otherwise, it gets -1 point. The fitness value is then the sum. The maximum value is 0.

2. What's your crossover operator?

   Pick an $1 \leq i \leq n$, generate

   $$[w_1, ..., w_i, w'_{i+1}, ..., w'_n, w'_{n+1}]$$

   from

   $$[w_1, ..., w_{n+1}], [w'_1, ..., w'_{n+1}]$$

   where w_{n+1} and w'_{n+1} are the thresholds of the respective perceptrons.

3. What's your copy operator?

   Tournament selection.

4. What's your mutation operator, if you use any?

   Not sure need any. I'd turn a weight from w to $-w$ to see how it works.

5. What's the size of the initial generation, and how are programs generated?

   I'd go with 10,000 minimum.

6. When do you stop the evolution? Evolve it up to a fixed iteration, when it satisfies a condition on the fitness function, or a combination of the two?

   I'd use a combination: stop when one of the program has max fitness value or when it reaches the MAX iterations.

7. What's the output of your system for the training set in the next page? This training set will be uploaded to canvas as a csv file.

   The training set was randomly generated from the following perceptron:

$$[0.1, -0.5, 1.2, -0.9, 2.3, 0.2, -2, -1.2, 1.1, 0]$$