IT3708 Module 4 - Iver Egge, Johan Slettevold - April, 2016

1 Implementation

1.1 Genotype and phenotype

The genotypes are stored in a bit vector, with 8 bits for each weight. These 8 bits represent the binary value of the phenotype. When converting the genotype to the phenotype, we first find the decimal value of the binary value, divide it by 2^8 and then scale that value between the desired minimum and maximum value of the phenotype

$$(n \cdot (max - min)) + min$$

which is then the floating point value of the phenotype.

1.2 CTRNN

The input to the CTRNN is composed as follows

$$Bias, T_{c1}, T_{c2}, x_1 ... x_n$$

Where *Bias* is the bias input, T_c is the time constraints. We use 5 values of x (the shadow-sensors of the agent) for the wrap-around scenarios and an addition two values (the wall-sensors) in the no-wrap-around scenario.

The time inputs T_{c1} and T_{c2} are calculated with the sigmoid variation provided in the assignment text, which is based on the previous neuron state for the respective neuron as well as a gain weight.

The weights of the first layer is then composed of a 8x2 or 10x2 matrix (based on the phenotype) which is multiplied with the inputs to get a 1x2 output matrix.

The values from the output matrix is run through a normal sigmoid activation function.

The input to the second layer is then composed similarly as

$$Bias, T_{c1}, T_{c2}, z_1, z_2$$

Where *Bias* and T_c calculated the same way as above, while z_1 and z_2 is the activated output from the first layer. This layer is then multiplied with the weights of the second layer and we get a 1x2 matrix.

This output matrix is run through a normal sigmoid function.

The two resulting values are the agents desire to go left and right. We map these numbers between predefined values for 0 to 4 steps, find which is largest and then send the agent in that direction.

Neuron states for the four neurons are updated for each forward propagation.

2 Performance of the EA

The fitness function is the same for all scenarios, only the bonus and penalty values are changed

To prevent agents from reaching a state where they either catch or avoid all blocks, we define a relation between capturing a small object and avoiding a big object. Since big objects are twice as rare to spawn, the *avoidedBig* value is multiplied by two.

$$Relation = |capturedSmall - avoidedBig * 2|$$

We continue by defining a score value which is composed by negative factors subtracted from positive factors. The four factors we use are avoided big objects, avoided small objects, captured big objects, and captured small objects; Each multiplied with a factor set in the configuration.

The fitness is calculated by dividing the score with the relation, however, the relation value may drop to zero. This is solved by editing this value if it is too low. To add some slack, we set the relation value to one if it falls under four.

2.1 Standard scenario

The agent waits under the object, sizing it up by moving slowly across the world. When searching it skips in larger steps. It has troubles with objects of size four, sometimes avoiding them completely, but generally performs well at avoiding big objects and capturing small objects. For this scenario we used the following values for bonuses and penalties:

$bigObjectBonus = 2.0$, $bigObjectPenalty = 0.0$, $smallObjectBonus = 1.0$, $smallObjectPenalty = 0.0$

2.2 Pull scenario

The agent behaves similarly to the standard scenario, except now it sometimes tries to pull the object so that it lands on part of the agents platform. It almost never waits for small objects to fall on their own when capturing. On big objects it waits until they fall on their own and does not utilize pulling. For this scenario we used the following values for bonuses and penalties:

$bigObjectBonus = 2.0$, $bigObjectPenalty = 2.0$, $smallObjectBonus = 1.0$, $smallObjectPenalty = 1.0$

2.3 No-wrap scenario

The agent moves semi-randomly around and seems to try to capture all objects possible, regardless of size. The agent changes direction when hitting the wall. For this scenario we used the following values for bonuses and penalties:

$bigObjectBonus = 2.0$, $bigObjectPenalty = 0.0$, $smallObjectBonus = 1.0$, $smallObjectPenalty = 0.0$

3 The evolved CTRNN

Evolved weights for the CTRNN in *standard* scenario:

$[w_{b1}, w_{b2}, w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}, w_{11}, w_{12}, w_{13}, w_{14}, w_{t1}, w_{t2}, w_{g1}, w_{g2},$
$w_{b1}, w_{b2}, w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_{t1}, w_{t2}, w_{g1}, w_{g2}] =$

[-3.882352828979492, -2.627450942993164,
-4.647058963775635, 3.117647171020508, 0.29411792755126953, 4.411765098571777, 5.0, 5.0, -0.6470589637756348,
-1.8627450466156006, -1.627450942993164, -0.5686273574829102, 1.8627452850341797, -2.6470587253570557, 1.3921570777893066,
2.215686321258545,
1.372549057006836, 1.929411768913269, 4.937254905700684, 1.2039215564727783,
-5.921568393707275, -3.9215683937072754,
-3.039215564727783, 5.0, 0.6470589637756348, -2.4509801864624023, -2.882352828979492, 3.7058820724487305,
-2.56862735748291, 2.2941179275512695,
1.2862745523452759, 1.941176414489746, 2.7725491523742676, 2.0352940559387207]

Analyzing the weights simply by observing them is no easy task. However, one could discover some trends of the agents by looking at the last weight layer (from hidden to output). These are the eight weights before the four last ones. As we can see, typically every second weight is larger than the previous (like the one that is of value 5.0). This suggest a movement primarily going to the right. This is not the case for the second pair, which could be because a detection here is relatively safe to stop at. A logic way to decypher this is that an agent not detecting anything at its back sensor, but at its following sensor, knows that it has found one edge of a block. If it also has found the front edge (inside its area of 5) it is sure that it should seize moving to the right.
The following shows how the agent acts based on some example inputs:

Input	**Output**	**Comment**
[0,0,0,1,1]	[[0.00012315571075305343, 0.2891501784324646]]	The agent moves to the right to explore the object
[0,0,1,1,0]	[[0.0005214650300331414, 0.0838247761130333]]	The agent stands still to capture the object
[1,1,1,1,1]	[[1.7498203305876814e-05, 0.771717369556427]]	The agent moves many steps to the right to avoid the object
[0,1,1,1,1]	[[0.0001611169136594981, 0.21411341428756714]]	The agent moves to the right to explore the object
[0,1,1,1,1]	[[0.07129821181297302, 0.00364687736146152]]	The agent stands still to capture the object

Table 1: CRTNN inputs and outputs

As seen from the last two rows, the agent has learned that the object is of size 4 by moving to the right and then to the left one step. It then waits to capture the object, having confirmed that the object is of appropriate size.