Sub symbolic AI methods: Project 4

Evolving neural networks for a minimally-cognitive agent

Bjørnar Walle Alvestad

2016

Norwegian University of Science and Technology

Implementation

Genotype and phenotype representation

The genotype for this system is represented by a series of bytes, each denoting a particular weight or parameter of the neural network. This representation grants 256 different values for each network parameter. The first n values in the byte sequence denotes the connection weights of the network, followed by the bias values, gain values, and then the time constants. When converted to phenotype, each parameter is scaled to fit the range [-1.0, 1.0] by dividing the two's complement binary value by 127. The values are then scaled to fit the correct range for the respective parameter category.

Parameter:	Range:
Weights	[-5.0, 5.0]
Biases	[-10.0, 0.0]
Gains	[1.0, 5.0]
Time const.	[1.0, 2.0]

CTRNN implementation

The neural network is implemented as an array of neural layers. Each layer accepts an input vector, which in turn is passed to every neuron in the respective layer. To accommodate for the intra-connected hidden- and output layers, each neuron also accepts a secondary input vector. This is automatically generated, and contains all the neuron output values from the last run for the respective layer. To produce an output vector, the layers are recursively activated, passing the output of one layer to the input of the next. The vector produced by the last layer is considered the output of the network.

Each component of the output vector represents an action performed by the agent. The performed action is determined by the highest output component. Experiments regarding activation thresholds were unsuccessful, and is not implemented in the final system.

EA performance

Standard scenario

In the standard scenario, the agent is able to capture all small objects, and is able to sense big objects and react by "fleeing" from the object. However, this is not always successful, as the tracker often wraps around the world and accidentally capture (or partially captures) the big object. For the small objects, the tracker oscillates back and forth under the object, waiting for it to capture. The performance of this agent could potentially increase if the tracker stopped completely and waited for the small object to capture (or big object to avoid). Although, the threshold functionality did not boost the performance.

The fitness function is based on a scoring system, rewarding the agent for performing desired actions, and deducted points when performing poorly. The score table is presented to the right, showing awarded credit for different situations.

	Big object	Small object
Capture	-0.5	+0.5
Avoidance	+0.5	-1.0
Neither	-0.5	-0.5

Pull scenario

In the pull scenario, the agent performs perfectly by capturing all small objects while avoiding all the big objects. It uses the pull action actively, by always pulling down small objects to achieve capture. For big objects, the tracker first scans the object to verify its size, then pulls the objects so that it always lands right behind the tracker itself, thus achieving avoidance. The tracker seem to constantly move in one direction.

The fitness function is similar to the scoring system used in the standard scenario, but stimulates usage of the pull action by rewarding extra credit when performed correctly. Correct usage is defined as:

- Capture performed by pulling small object. (+0.3 credit).
- Avoidance performed when pulling big object. (+0.9 credit).

This is credit is rewarded as extra bonus to the base score.

No-wrap scenario

This scenario performs similar to the standard scenario, constantly moving in one direction until the edge is reached. The direction is then reversed. When approaching small objects, the tracker starts the oscillation movement described in the standard scenario section. Because of the no-wrap policy, the agent may fail to avoid big objects close to the edges, as the tracker is not able to escape to the other side of the map. This can possibly be prevented by having the tracker try to escape in the opposite direction, but for this to work, the agent must travel to the other side of the object. This is a hard behavior to evolve.

The fitness function used in the no-wrap scenario is similar to the score system introduced in the standard scenario. The awarded/deducted credits are modified to some extent. The new score-board is shown to the right:

	Big object	**Small object**
Capture	-0.5	+0.5
Avoidance	+0.7	-1.0
Neither	-0.8	-0.5

CTRNN analysis

Weights and behavior analysis

The following table shows the evolved weights for a standard agent.

W	[-1.73, -4.61, 2.72, 3.82, 1.26, -1.93, 1.46, 4.21, 4.45, -2.52, 3.19, -3.58, -0.47, -1.81, 0.51, 4.84, 1.34, -2.24, 1.38, 3.23, 4.80, 1.69, -0.98, -3.27, 4.53, 4.96, -2.36, -0.94, 3.11, 4.17, -2.76, 4.92, 3.03, 2.28]
B	[-2.13, -0.08, -4.09, -4.33, -6.34]
G	[2.31, 3.11, 2.62, 4.29, 1.06]
T	[1.00, 1.87, 1.48, 1.63, 1.50]
W = weights (connections and recurrence), B = biases, G = gains, T = time constants	

When running the network, the ordinary neuron connection weights determine the agent behavior in one discrete environment. The output should be equal for equal input, but the recurrence weights introduce memory to the network. It is no longer guaranteed that the network will output the same result for equal input, because the neurons has states of saved activation. This is reflected in behavior of the agent. When sensing a big object, the agent “remembers” that a big object is present, and will try to flee from it.

The memory of the network can be demonstrated by feeding the same input multiple times. When feeding the empty sensor vector [0, 0, 0, 0, 0], the network produces [0.005, 0.005] on the first run, then [0.003, 0.001] on the second, and finally [0.004, 0.000] on the third. This reflects that the agent should move to the left, in hope to find a small object to capture. Another example is to “fool” the agent to believe there is a big object right on top of it. On the first run, the agent produces the output [0.010, 0.108]. On the second run however, using the exact same input vector, the network outputs [0.999, 0.354]; a strong indication to move to the left, out of harms way.