IT3708: Project 4 – Evolving Neural Networks for a Minimally-Cognitive Agent

Author: Eva Tesarova, Petr Zvonicek

a) Implementation

Genotype / Phenotype representations

Genotype is a bit vector and is divided into 8-bit chunks representing one decimal number (0 - 255). This decimal number represents one weight – one item of the phenotype. The genotype - phenotype conversion is done by converting the binary number to decimal, linear scaling (decoding) of the [0, 255] interval into the desired interval of the weight (eg. [-5, 5]). The phenotype - genotype conversion is slightly more complicated, it requires finding the closest value in the scaled values, encoding this value to [0, 255] interval and converting this to binary.

CTRNN

The CTRNN is the subclass of the network from Flatland project. In addition we added to neurons attributes (bias weight, gain, time constant) needed for CTRNN output computation.

The motor output is formed by two neurons, which gives output in the range from 0 to 1. The tracker direction is decided by the neuron which gives higher value (i.e. one represent *right* direction, one *left* direction. The speed of the tracker is then computed from the value of the "winning" neuron. By multiply this value by 5 we scale its output to the 4 possible speeds.

With pull extension we gave to output layer one more neuron. It has the same properties as the others in this layer, i.e. it is connected to bias and also it to every neuron in this layer (self included). We interpret its output (which is again in range $(0, 1)$) as make a pull action if it is higher than the value 0.5.

b) Performance

Standard scenario

In the standard scenario, agent always choose at the beginning one direction which then hold for all the simulation. Its behaviour under the objects is various, depending on run of our EA. Two main approaches can be seen:

- It is going fast and first run through the world it "scans" the object and in the second it stops under if it is small object.
- It is going slower and when it is under object it tries to catch it at the end part. If it sees that object is big, it just pass without stop under it.

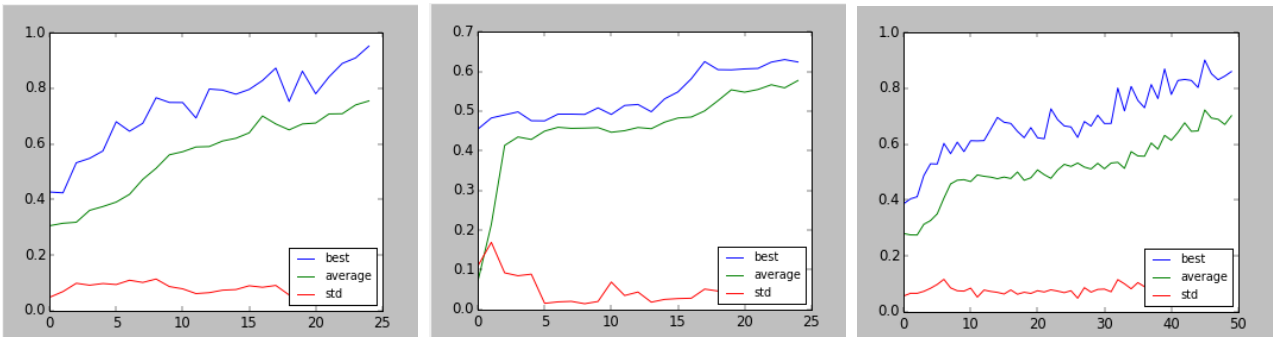
Our trackers has problems with catching objects while they appear immediately above them after they caught something. Usually they just continue with the rest and don't try to avoid/catch object.

Pull scenario

With the pull scenario agent behaves similarly as in the standard case. When it just stopped in the standard case and waited for the object, now he use pull action to get the object faster. Sometimes immediately after it takes the pull action it use it also for bigger object. On the figure is shown how EA first learn capture objectsm then the second graph jump captures learn use pull action.

No-Wrap scenario

This scenario is probably the most difficult for our EA, because evolving rational behaving agent needs more luck. With this scenario we added to the input sensors two more sensors detecting walls (on the right and on the left). We also make weights from this inputs twice bigger as the rest weights. We managed to educate agent which can turn after hit the wall as well as capturing objects (it was not able to avoid bigger ones). However, educating good agents demands many runs of EA and it is very sensitive to initial conditions, so we change fitness function to help agents to learn turn after hit wall. When some agent turns, he is given a big reward for this behaviour. He can obtain this reward only few times per its live, since we want to avoid hitting only wall all the time. This approach was showed to be good since agent learns both capturing objects and turn after hit wall. The develop is shown on the picture, where is perceptible two jumps, first turn from the wall, second capture objects.



c) CTRNN Analysis

We use the same topology as was in the assignment. Lets mark neuron from the input layer as $i_1, \ldots, i_5$, neurons from the hidden layer as h_1, h_2, neurons from the motor output layer as o_1, o_2 and the bias neuron as b. One of the evolved ANN weights are followed:

to\from	i_1	i_2	i_3	i_4	i_5	h_1	h_2	o_1	o_2	b
h_1	5.0	-4.647	-4.058	-1.274	5.0	2.019	2.294			-1.019
h_2	-3.862	0.098	3.980	-2.960	5.0	1.352	3.0			-0.352
o_1						2.176	-1.0	1.0	-3.0	-0.705
o_2						-3.0	4.294	1.0	5.0	-9.019

	h_1	h_2	o_1	o_2
gains	4.921	3.996	2.003	2.992
time constants	1.0	1.250	1.003	2.0