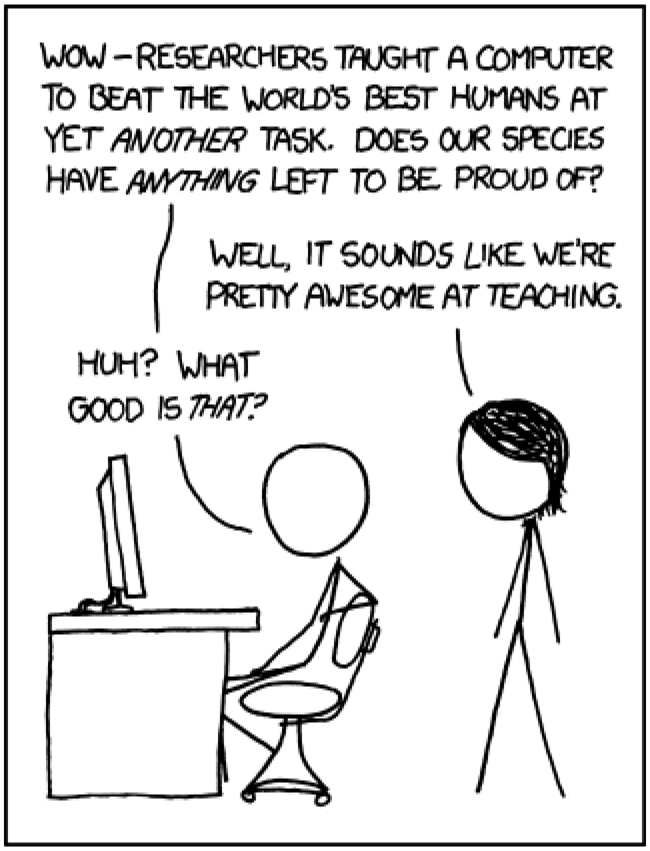
ECTA Homework 5
Neuroevolution with the Enforced Subpopulations (ESP) algorithm

Djordje.vukcevic, Supriya Vadiraj
djordje.vukcevic@smail.inf.h-brs.de, supriya.vadiraj@smail.inf.h-brs.de

July 12, 2018


WOW – RESEARCHERS TAUGHT A COMPUTER TO BEAT THE WORLD'S BEST HUMANS AT YET *ANOTHER* TASK. DOES OUR SPECIES HAVE *ANYTHING* LEFT TO BE PROUD OF?
WELL, IT SOUNDS LIKE WE'RE PRETTY AWESOME AT *TEACHING*.
HUH? WHAT GOOD IS *THAT?*

1 Assignment Description

1. Solve the following Reinforcement Learning tasks with Neuroevolution.
   - (a) Single Pole Balancing with Velocities
   - (b) Double Pole Balancing with Velocities
   - (c) Double Pole Balancing without Velocities
2. Compare a standard approach to the ESP algorithm in (c).

- The task is "solved" if the poles does not fall for 1000 time steps.
- A 1 and 2 pole balancing simulator function is attached. Given a state and a command it computes the state at the next time step. Examples are included (help: onePole, twoPole).
  Note: there is only one 2 pole balancing function, in the case without velocities it is up to you to "blind" you ANN of these states, but they are required by the simulator to compute what happens in the next time step.
- You are responsible for all other code, including the ANN.
- Additional resources for ESP are available on the LEA

2 Submission Instructions

Follow along with the instructions in this PDF, filling in your own code, data, and observations as noted. Your own data should be inserted into the latex code of the PDF and recompiled. All code must be done in MATLAB.

To be perfectly clear we expect two submissions to LEA:

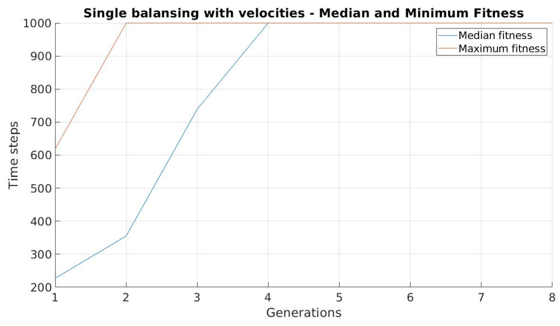
1. 1 PDF (report) – a modified version of your submission PDF, with your own code snippets, figures, and responses inserted
2. 1 ZIP (code) – a .zip file containing all code use to run experiments (.m files) *and* resulting data as a .mat file
3. 4 MAT (solution) – Mat files with the weight matrices of the best found solutions: *spb.mat*, *dpb.mat*, *dpbnv.mat*, and *dpbnv_esp.mat*

3 The Assignment

3.1 Single Pole Balancing (25pts)

Implement a feed forward network which solves the single pole balancing problem using a fixed topology and a GA or ES (your choice).

- (15 pts) Describe your approach. Be sure to mention: the topology of your network, the form of the genotype, the form of the phenotype, the optimization approach used and its details including hyperparameters. Your results should be replicable from your description. Pictures are worth 1000 words.
- **Answer** : Optimization Approach: We used CMAES with evolutionary paths to train our neural net. The toplology contains 3 hidden nodes without bias, the number of inputs were set to 4. The chromosome contains weights which describe connections from input nodes to all hidden nodes + connections from hidden nodes to all output nodes. We used a feed forward neural network.
  - Hyperparameters for CMAES:
  - Initial sigma = 0.3
  - Max no of generations = 700;
  - Stop criteria = neural network with fitness 1000 time steps.
- (5 pts) Plot the fitness progress over 10 runs.

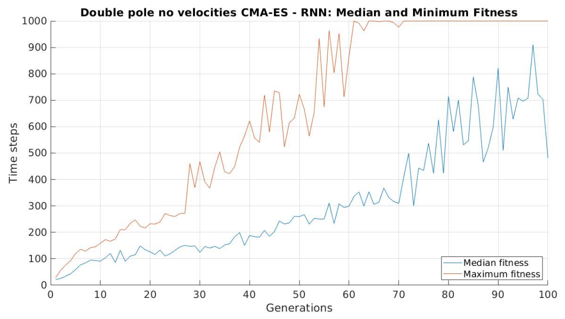


- (5 pts) Upload the weight matrix of your most successful solution in a file called: *spb.mat*
- The solution was found in generation 7.

3.2 Double Pole Balancing with Velocities (25pts)

Adapt your feed forward network to solve the double pole balancing problem.

- (15 pts) Describe how this approach differs from above. Your results should be replicable from the this description and the one above.
- **Answer** : Optimization Approach: We used CMAES with evolutionary paths to train our neural net. The toplology contains 3 hidden nodes without bias, the number of inputs were set to 6. The chromosome contains weights which describe connections from input nodes to all hidden nodes + connections from hidden nodes to all output nodes. We used a feed forward neural network.
  - Hyperparameters for CMAES:
  - Initial sigma = 0.3
  - Max no of generations = 700;
  - Stop criteria = neural network with fitness 1000 time steps.
- (5 pts) Plot the fitness progress over 10 runs.



- (5 pts) Upload the weight matrix of your most successful solution in a file called: *dpb.mat*
- The solution was found in generation 21.

3.3 Double Pole Balancing without Velocities (50pts)

Alter your network topology to solve the double pole balancing problem without velocities using a recurrent neural network. Implement the ESP algorithm to optimize the same network topology. Compare the performance of the two approaches.

- (10 pts) Describe how the first approach differs from above. Your results should be replicable from the this description and the one above.
- For solving this task using CMAES algorithm we change our neural network to be recurrent and this led to larger size of chromosome due to additional weights added.
- Hyperparameters for CMAES with RNN:
- Input size = 3
- Rest is the same as above.
- (25 pts) Describe the ESP approach, how it differs from above, and other details, including hyperparameters.
- **Answer** : Optimization Approach: We used ESP (Evoutionary Sub population) to train our neural net. We used burst mutation to accelerate the search for the optimum weights. The chromosome contains weights which describe connections from input nodes to all hidden nodes + connections from hidden nodes to all output nodes + recurrent connections between each hidden node and the nodes itself. We used a recurrent neural network. In this case we had to optimise more weights.
  - Hyperparameters for ESP:
  - mutation_prob = 0.4
  - subpopulation size = 40
  - Max no of generations = 800;
  - Stop criteria = neural network with fitness 1000 time steps.
  - generation limit for burst mutation = 20
- (10 pts) Plot the fitness progress of both approaches over 10 runs, including significance
- (5 pts) Upload the weight matrix of your most successful solution in 2 files called: *dpbnv.mat* and *dpbnv_esp.mat*
- The solution was found in generation 20 for CMAES without velocities.
- The solution was found in generation 528 for ESP without velocities.

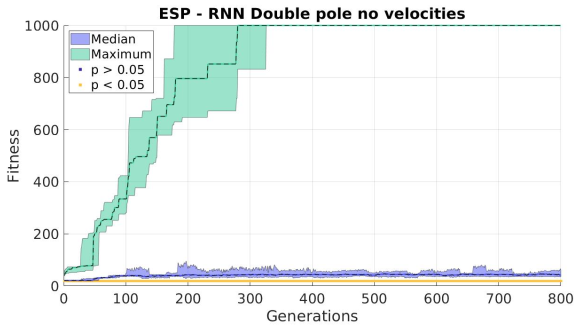


Figure 1: double_pole_no_vel_ESP_RNN

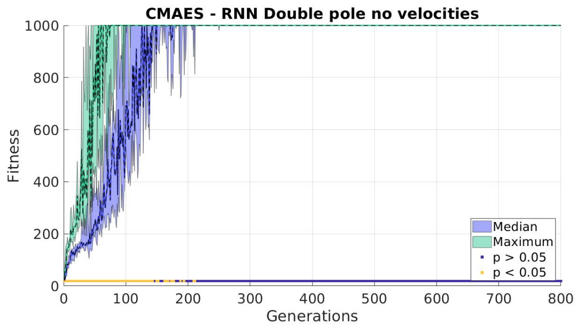


Figure 2: double_pole_no_vel_cmaes_RNN