Alteration the observation state of an agents for achievement of the desired behavior.

Dmitrii Desiatkin

d.desyatkin@innopolis.university

Introduction

Reinforcement learning is a powerful instrument that allow to solve a various set of tasks. From creating new bots for popular games like chess, to creating personal recommendation systems. From real robot control to image processing. The reason for that is huge adaptivity of such systems, they can learn even changing game rules like was demonstrated in the paper (Mnih, Silver, and Riedmiller, n.d.). However in that work I am interested in application of such algorithms to simulations of the real robots declared in gym and pybullet environments.

I want to control an agents with simple neural networks with inputs of agents sensors. Also I would not use any gradient decent and other strict computational algorithms for training of that networks. Instead I decided to use evolutionary algorithms for networks weights updates. I will use direct evolution method described by (Pagliuca, Milano, and Nolfi 2018).

The main idea of that work is to check the modern approach introduced in the work of (Freeman, Metz, and Ha 2019). They introduced the idea of observational dropout, what proposes to force the agents to predict it's own future state without any additional information (like dynamic model of a robot, or additional information from sensors outside of an agent). To achieve that behavior authors suggest to hide the observation vector of an agent in particular moments of time, and use some part of the current output as input for the next time step.

Experimental setup

For experiments I have used the library developed by Stefano Nolfi and Paolo Pagliuca 2. It provides the agent models and various instruments for conducting the evolutionary experiments. Also they provided modifications for gym and bullet libraries, cause default reward functions do not allow to conduct the evolutionary experiments.

As targets I have used the xant locomotor and xdpole agent. Their default settings stated in the table bellow. All experiments are conducted with simple fully connected network architecture.

xdpole	xant
[ADAPT] maxmsteps = 10000 environment = ErDpole stepsize = 0.01 noiseStdDev = 0.02 sampleSize = 250 wdecay = 0 sameenvcond = 1 saveeach = 1 [POLICY] ntrials = 10 nttrials = 500 maxsteps = 1000 nhiddens = 10 nlayers = 1 bias = 1 out_type = 2 architecture = 3 afunction = 3 winit = 0 action_noise = 0 normalize = 0 clip = 0	[ADAPT] Maxmsteps = 50 environment = AntBulletEnv-v0 stepsize = 0.01 noiseStdDev = 0.02 sampleSize = 20 saveeach = 1 wdecay = 1 [POLICY] ntrials = 1 nttrials = 3 maxsteps = 1000 nhiddens = 50 nlayers = 1 bias = 1 out_type = 3 architecture = 0 afunction = 2 winit = 1 action_noise = 1 normalize = 1 clip = 1
Neural network schemes:	
3 → 10 → 1	28 → 50 → 8

Table 1: Original agents parameters

`https://github.com/snolfi/evorobotpy`

Xdpole environment

That environment represents 2d space with the two inverted pendulums fixed on the mobile cart. Observation vector represents the position of the cart on x axes and two angles (θ_1, θ_2) what represents the displacement of pendulums from upward position. Action space is one dimensional, it is the direction of applying constant force to the cart. Experiment was interrupted when the position of the cart exceeded range of $[-2.4, 2.4]$, and theta angles exceeded range of $\left[-\frac{\pi}{5}, \frac{\pi}{5}\right]$. More detailed description of the environment can be find in paper (Pagliuca, Milano, and Nolfi 2018)

Xant environment

That environment represents 3d space with four leged robot. Observation space consists of robot body current z position displacement from the initial one, sinus and cosinus of the angle to the goal, xyz velocities of the body, roll and pitch orientation angles of the body, current angular position and speed of eaach joint, and touch sensors at the end of each leg.
Action space has eight dimensions, they represents the torque what should be applied to each joint.

Modifications

For testing the hypothesis the common behavior of the agents was altered in the following way, output of controlling neural network was increased on the size of observation vector, values of all additional neurons was stored to be used on the next iteration either as sum with the previous observation or as additive component to the current input.
Also hidden state was increased for introducing additional variance into the model.

xdpole	xant
Nhiddens = 10 / 18 / 27 / 40 architecture = 0	Nhiddens = 50 / 90 stepsize = 0.04 noiseStdDev = 0.08

Table 2: Agent parameters modifications

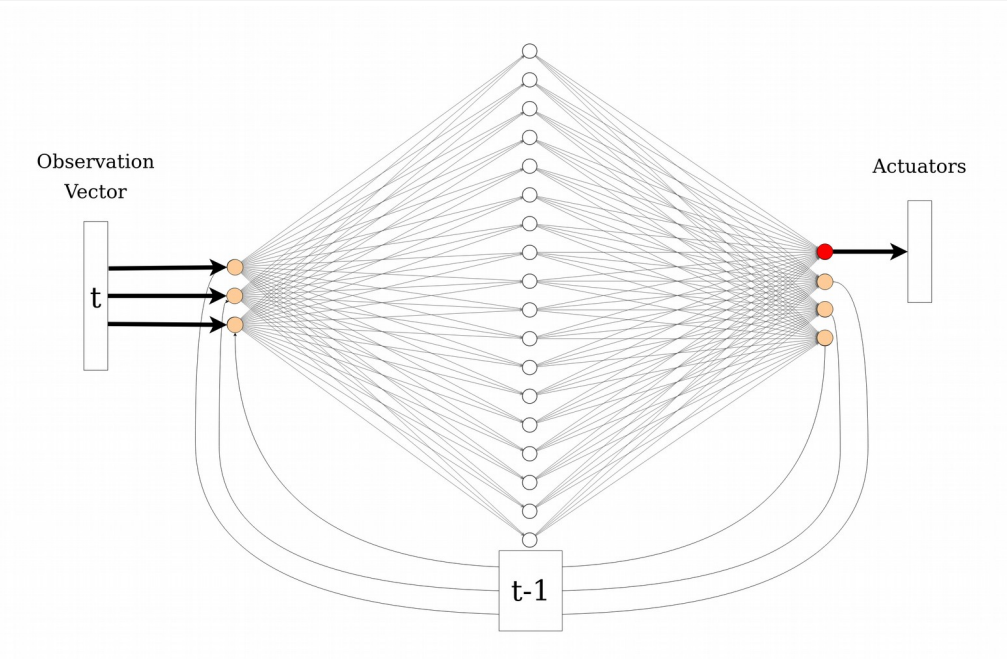


Illustration 1: The first variation of NN control graph for xdpole agent (18 neurons in hidden layer). We can see how predictions made on moment of time t-1 passed to actual time.

Variation 1	Variation 2
Input is the sum of real observation and some additive component calculated by network.	**Input is the pure observation in usual case, and the sum of observation in the moment t - 1 and some additive component calculated by network in case of hidden output.** **Proportion of hidden/usual is 20/80 for xdpole, and 80/20 for xant.**

Project code accessible from github:

https://github.com/d-desiatkin/Behavioral_robotics/tree/master/Project

Results

Wrong results

Bellow you will find the graphs what I acquired during very first experimentation on the stated task.

From graphs below I have done the conclusion that there are exist a bug in my code. That helped me to deeper understand the particular implementation of the evolutionary algorithms used by Stefano Nolfi and Paolo Pagliuca. I have studied that code in the beginning of the course and was assured that I remember it pretty good. However, I have mistaken. Bug appeared from incorrect saving of the previous output of the network. I have passed the output of one network to the next network within the same generation. Moreover I have never cleared that output, and had never checked that a new generation have started. What means that I have passed the output of the last network of previous generation, to the first network of new generation.

Also I must mention that graph plotted below may have small errors in the beginning and at the end. It happened because it was hard to distinguish the end of one experiment from the beginning of another. However I may guarantee that this mistake is no more than 20 points.

Because I have understood that code is wrong I did not run any experiments with xant in that environment.

I decided to save these results just because that wrong implementation allowed to solve agent’s tasks.

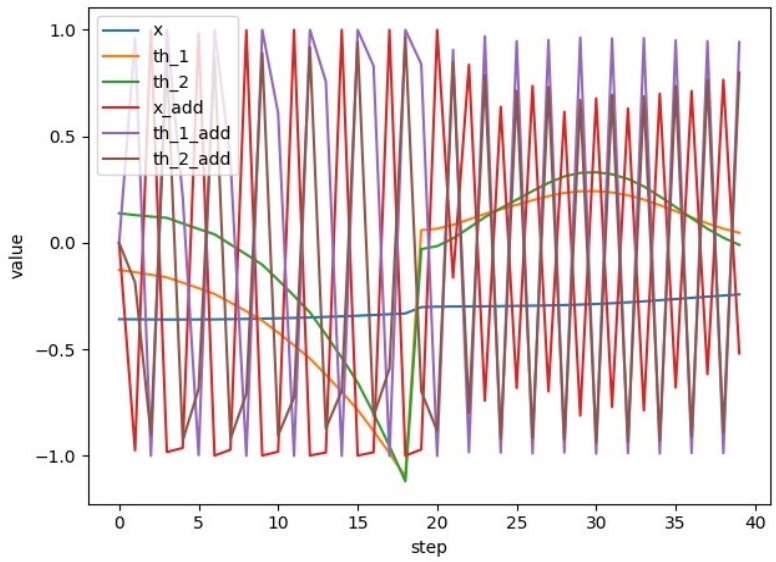


Illustration 2: Modified xdpole failure example. Variation#1. I can see the problem in my code on that graph. Theta variables exceded allowedboundaries and new experiment started. Although, old predicted variableswere passed to a new experiment. (nh = 18)

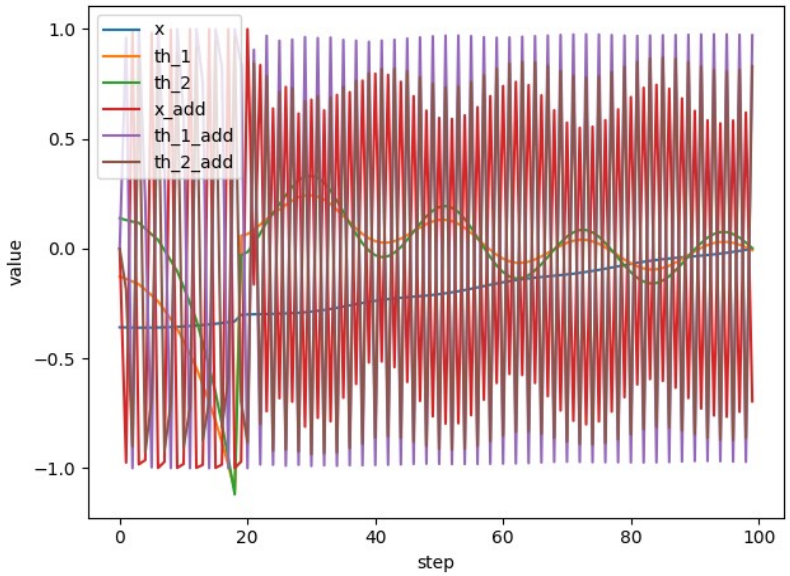


Illustration 3: Modified xdpole success example. Variation#1. Despite the problem shown on previous graph (Illustration 2), I can see that modification allowed stabilize xdpole. (nh = 18)

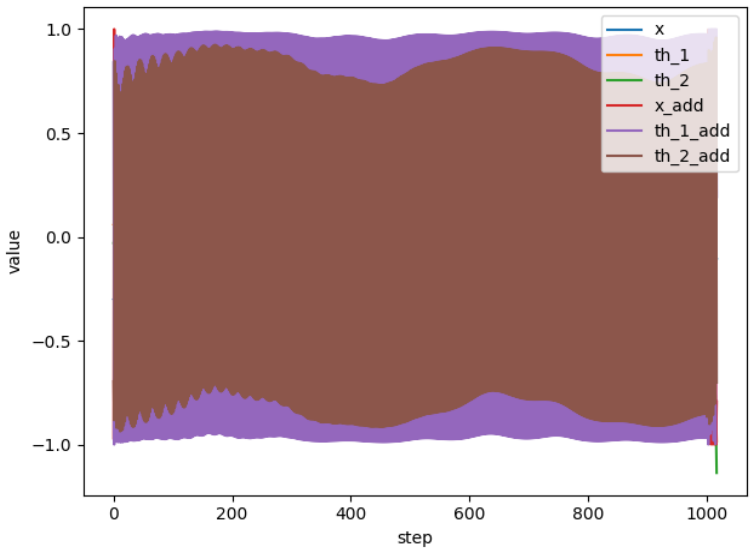


Illustration 4: Modified xdpole success example. Variation#1. Full experimentwhat started on the Illustration 3, I can see that th_1_add, and th_2_add,always near maximum values. (nh = 18)

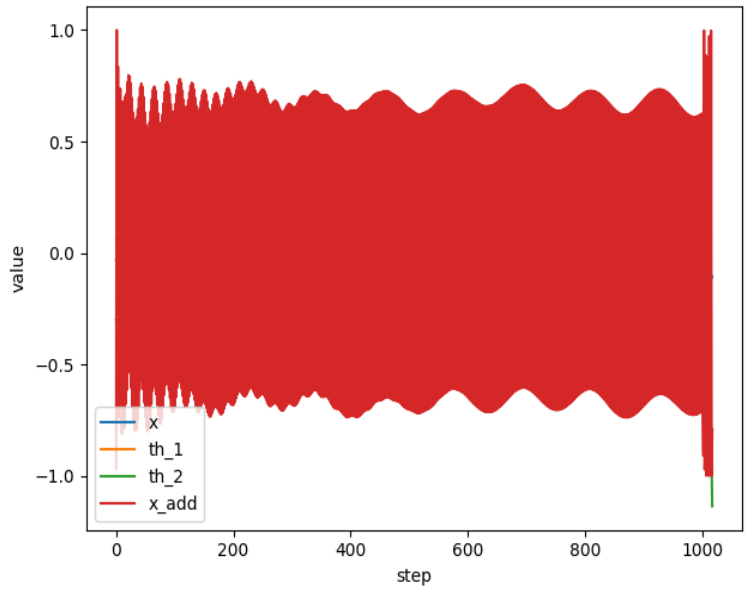


Illustration 5: Modified xdpole success example. Variation#1. x_pred behaviour.(n_hidden = 18)

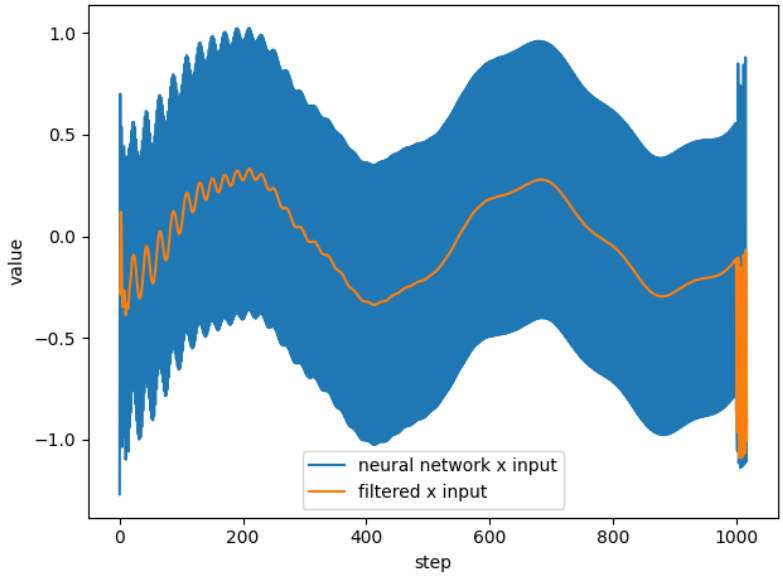


Illustration 6: Modified xdpole success example. Variation#1. Neural nework x (cart position) input signal (oservation + prediction), filter is rolling averagewith window size w = 2. (n_hidden = 18)

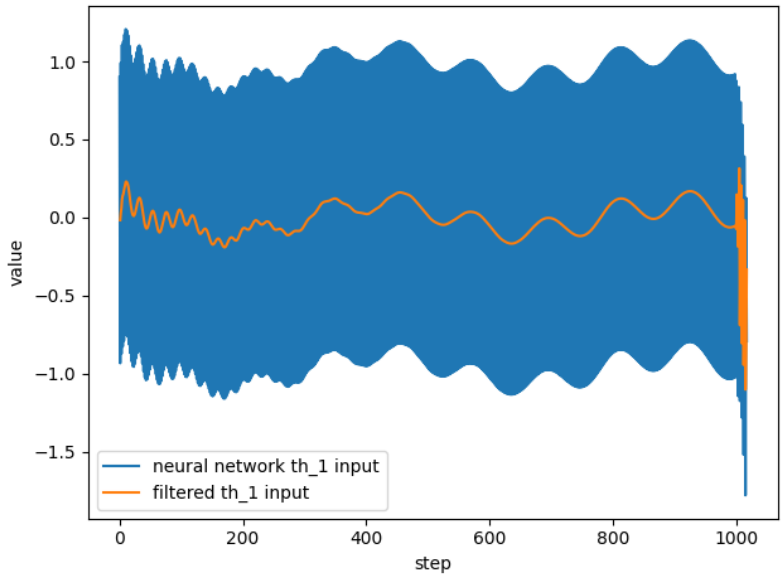


Illustration 7: Modified xdpole success example. Variation#1. Neural neworkth_1 input signal (oservation + prediction), filter is rolling average with windowsize w = 2. (n_hidden = 18)

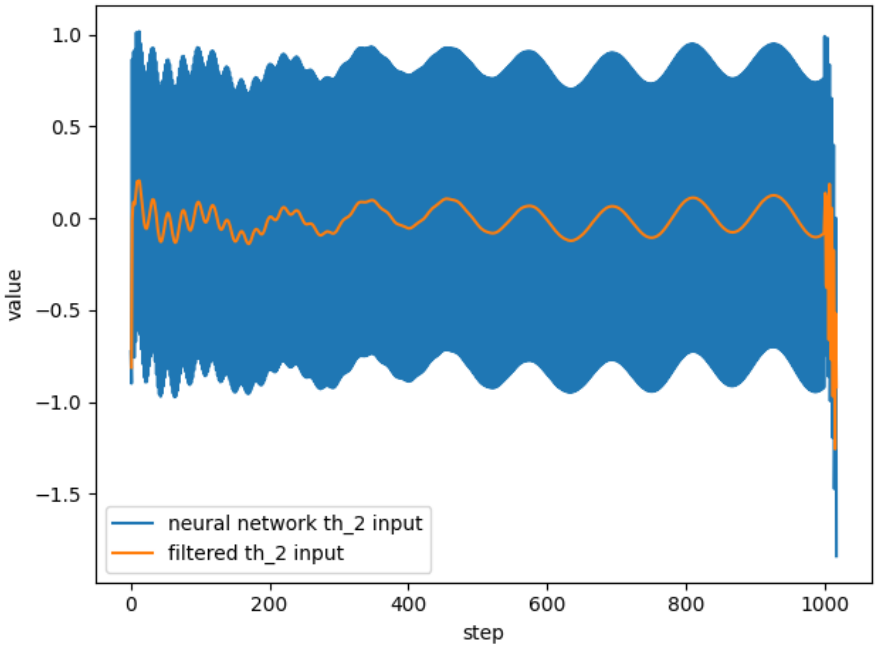


Illustration 8: Modified xdpole success example. Variation#1. Neural neworkth_2 input signal (oservation + prediction), filter is rolling average with windowsize w = 2. (n_hidden = 18)

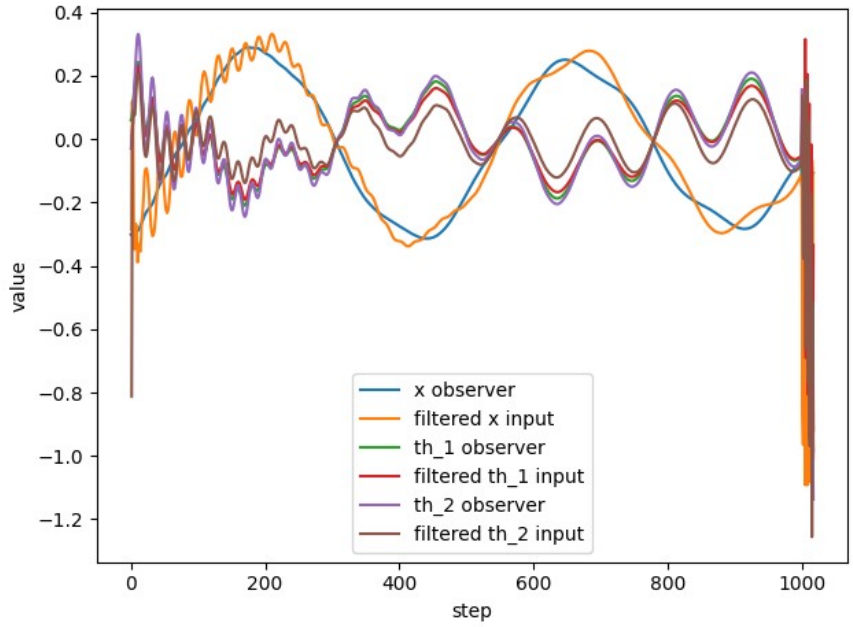
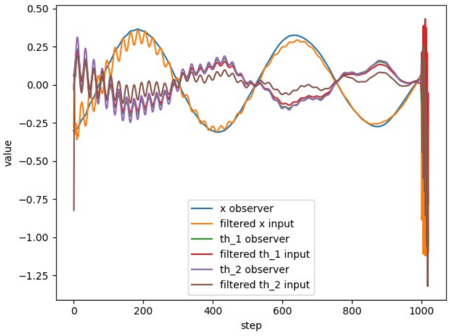
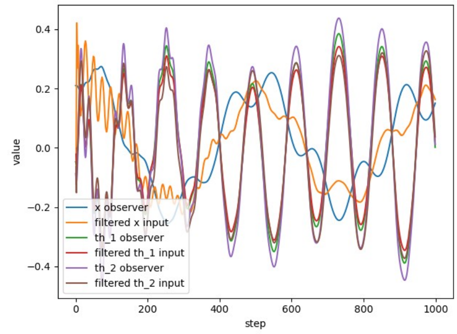


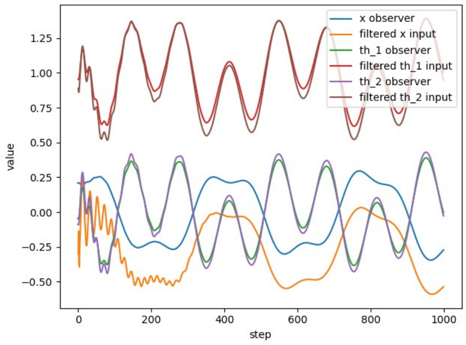
Illustration 9: Modified xdpole success example. Variation#1. Pure observationvector values vs. filtered modified input.

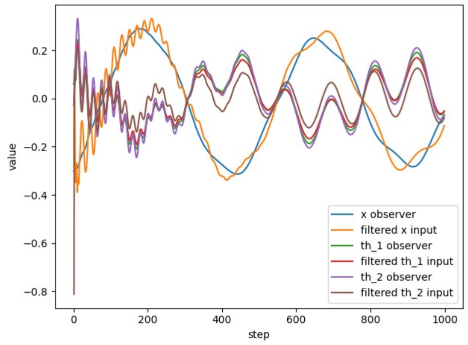
Illustration 10: Xdpole, nh=18, different seeds comparison

Best g fit

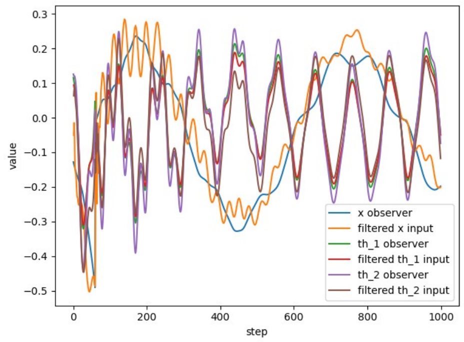


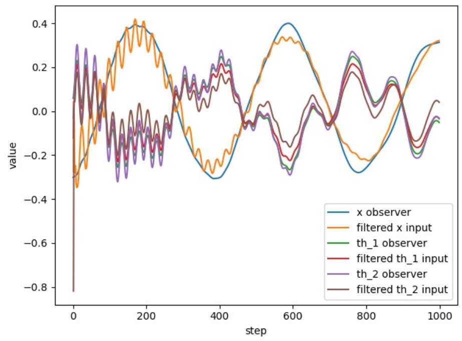


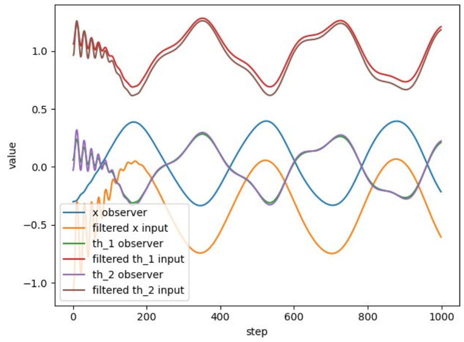




Best fit







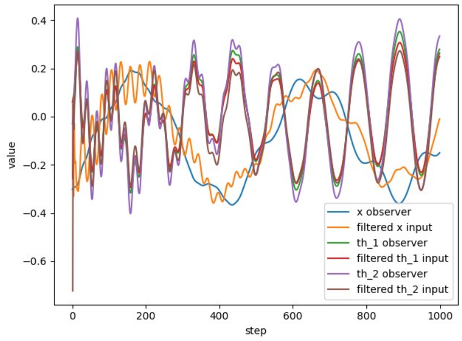
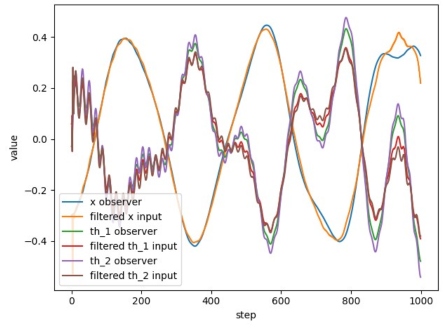
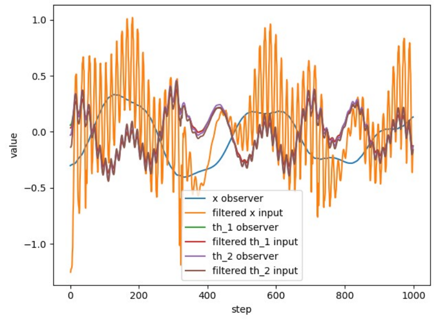
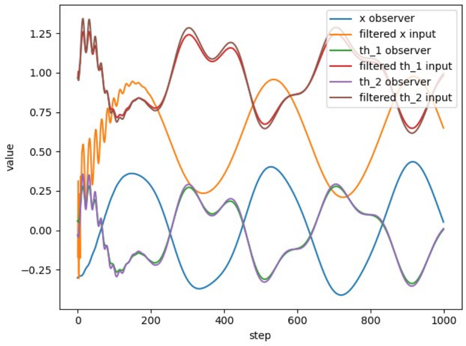


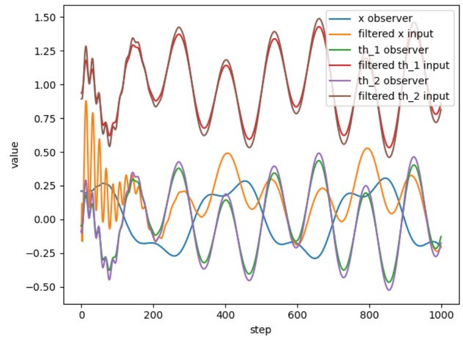
Illustration 11: Xdpole, nh=27, different seeds comparison

Best g fit.

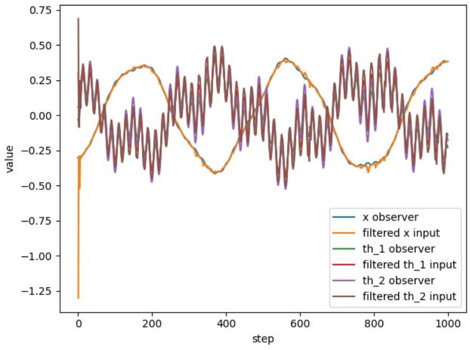


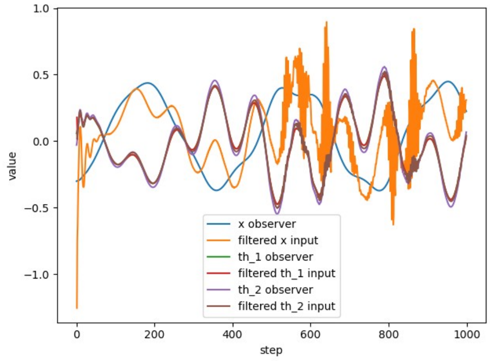


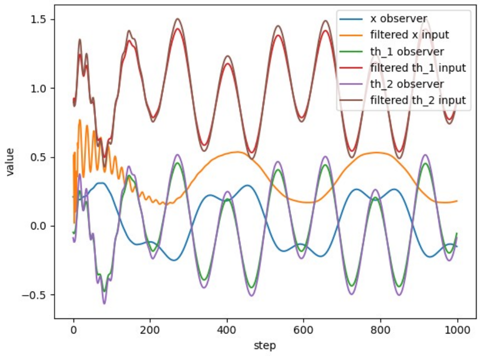




Best fit.







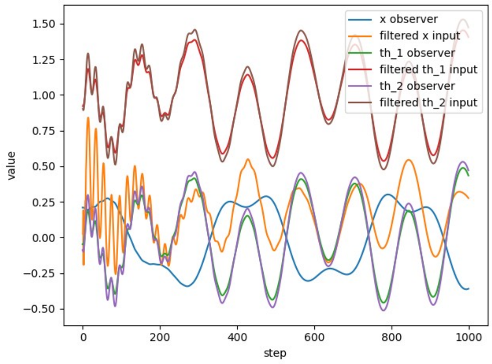
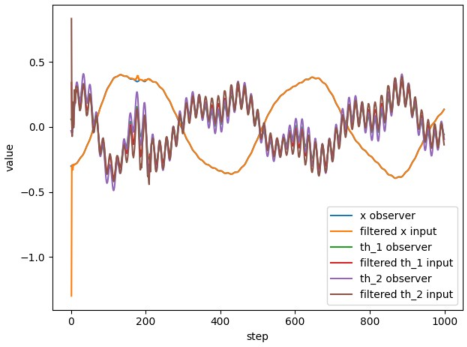
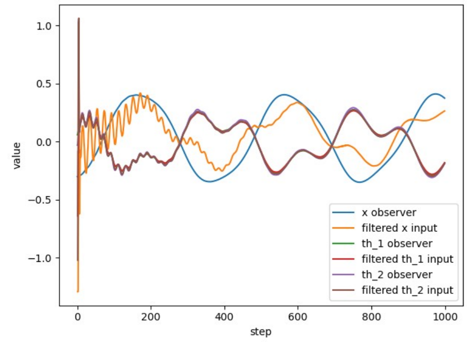
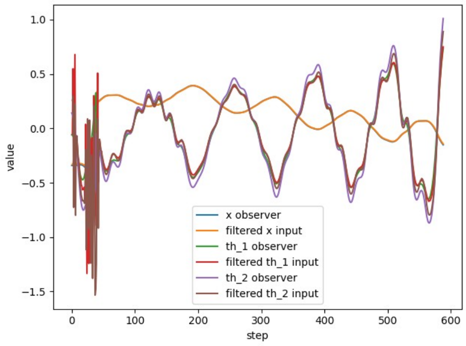


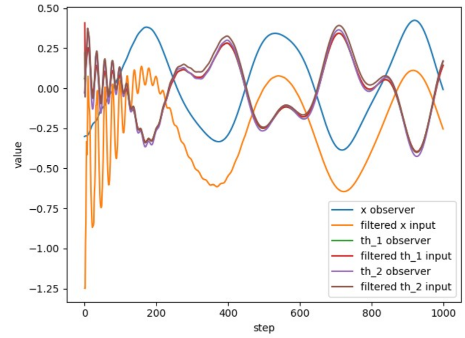
Illustration 12: Xdpole, nh=40, different seeds comparison

Best g fit.

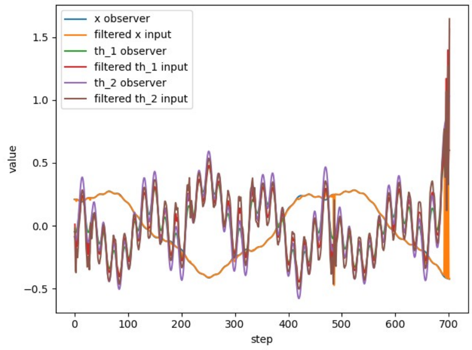


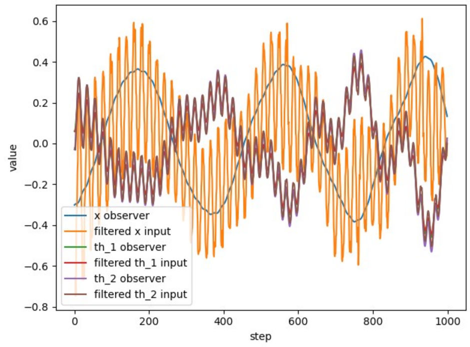


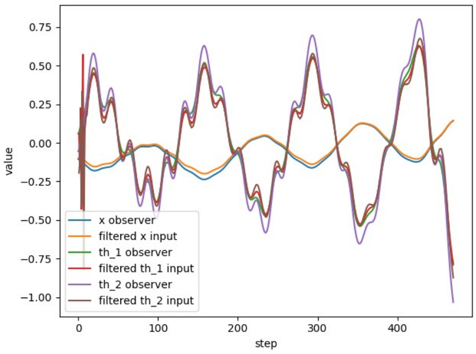




Best fit.







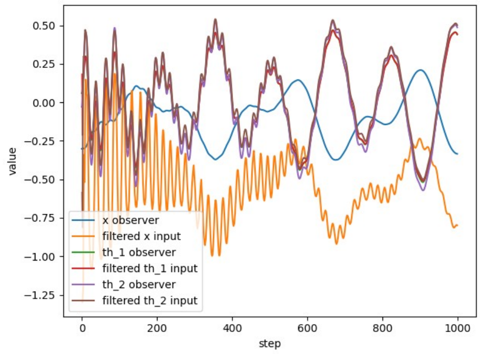
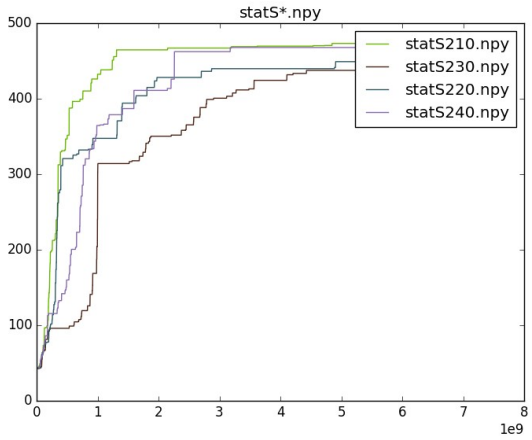
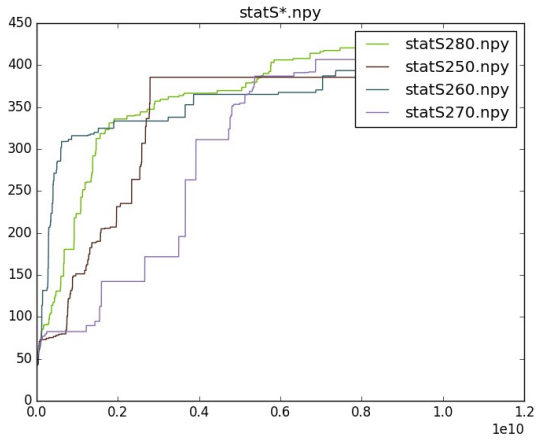


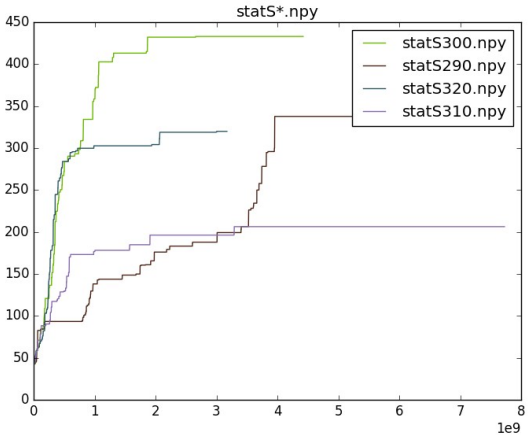
Illustration 13: Training process of xdpole



xdpole nh=18



xdpole nh=27



xdpole nh=40

First variation results

Bellow you will find the valid results for first variation. I have not enough computational powers to perform a lot experiments, that is why only 3 seeds for xdpole, and 2 seeds for xant was computed.

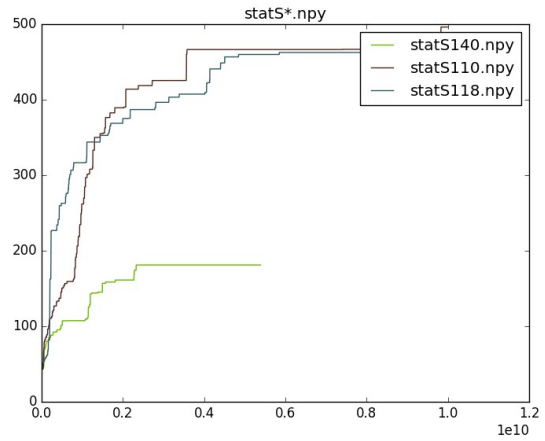
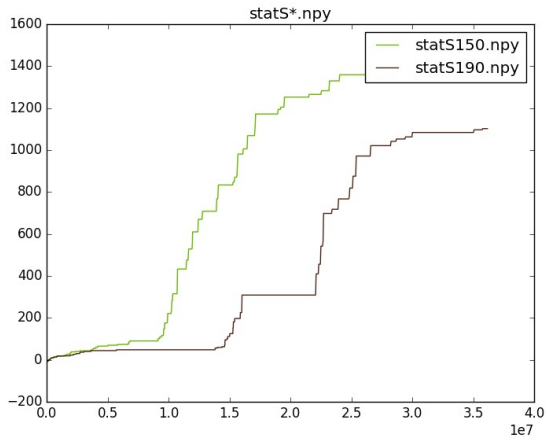
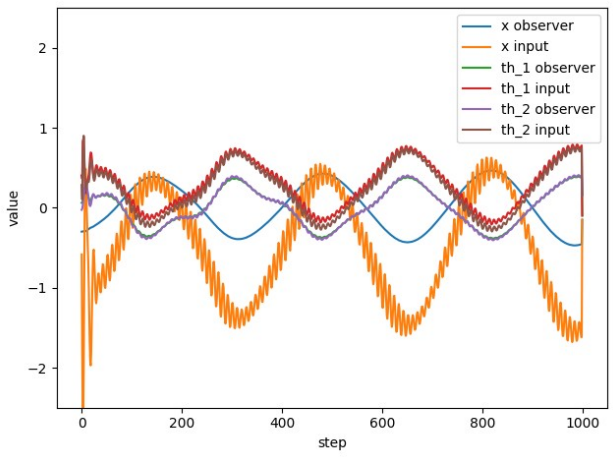
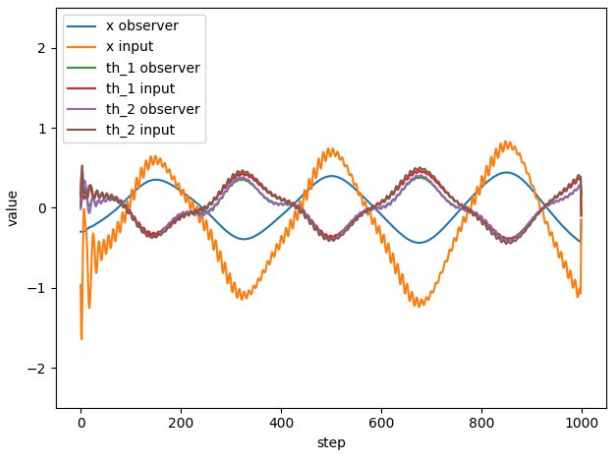
Illustration 14: Training process of xdpole and xant	
	
xdpole training process for different number of hidden neurons (seed - 100 corresponds to number of hidden neurons)	xant training process for different number of hidden neurons (seed - 100 corresponds to number of hidden neurons)

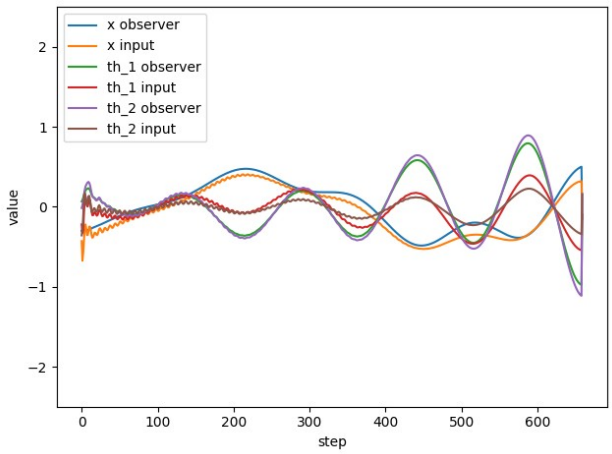
Illustration 15: Comparassion of observation vector vs NN input for xdpole environment. Best generation sample



nh = 10

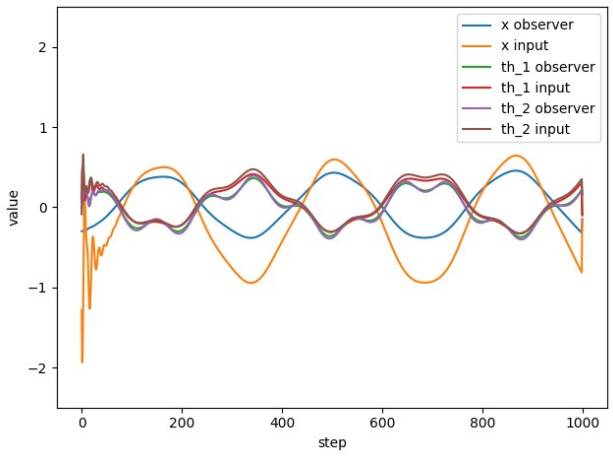


nh = 18

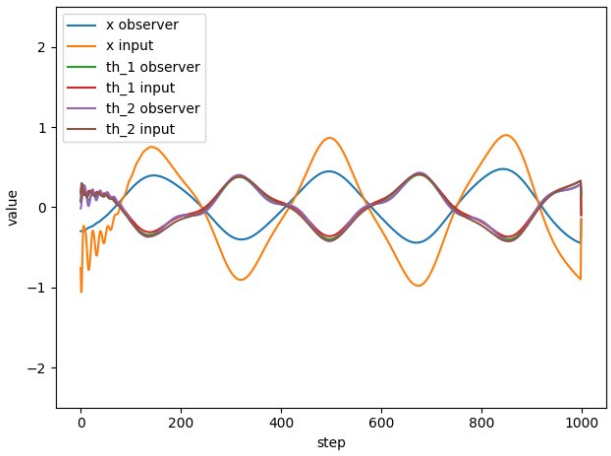


nh = 40

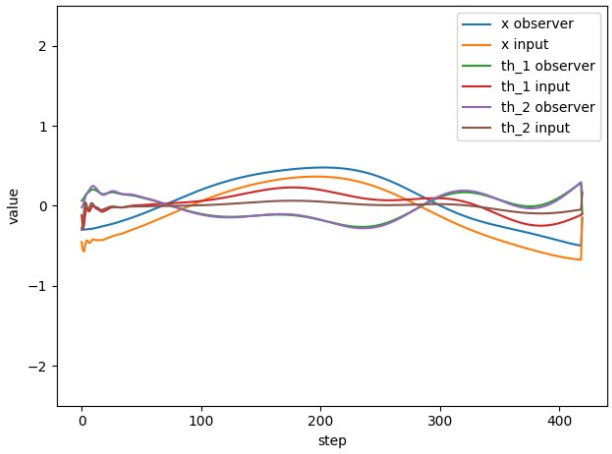
Illustration 16: Comparassion of observation vector vs NN input for xdpole environment. Best sample



nh = 10



nh = 18



nh = 40

Second variation results

Bellow you will find the valid results for second variation. Because I have not enough computation powers for such small time amount, here also only several experiments was conducted.

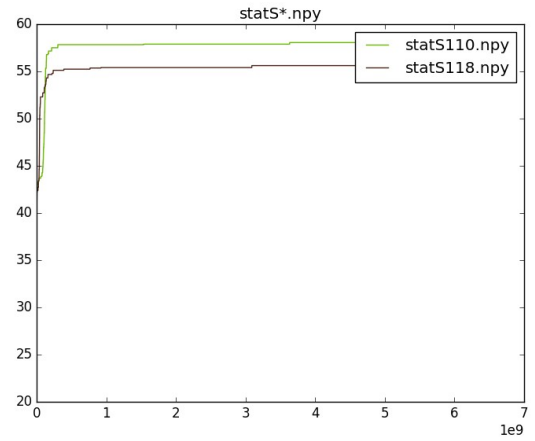
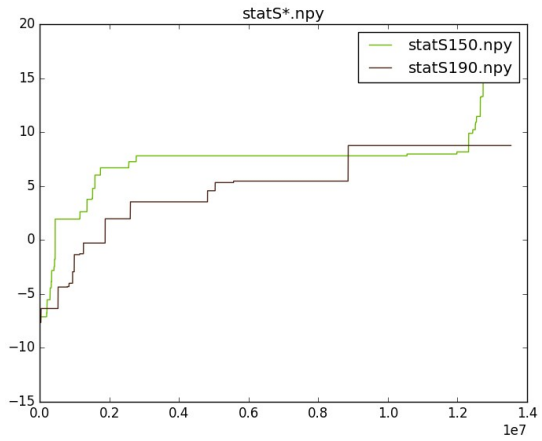
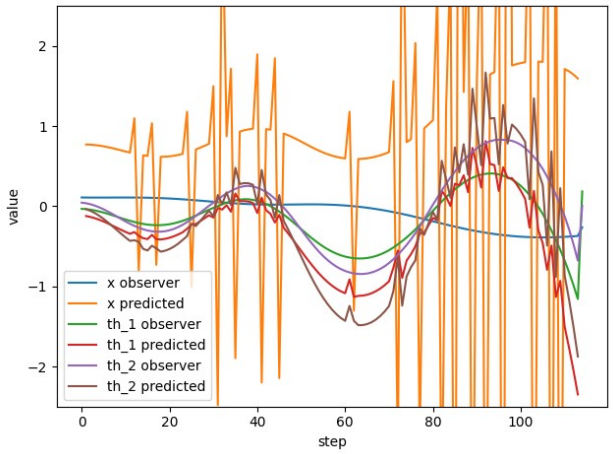
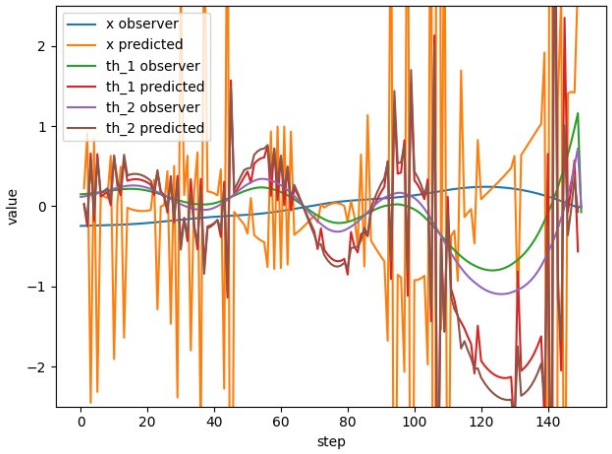
Illustration 17: Training process of xdpole and xant	
	
xdpole training process for different number of hidden neurons (seed - 100 corresponds to number of hidden neurons)	xant training process for different number of hidden neurons (seed - 100 corresponds to number of hidden neurons)

Illustration 18: Comparassion of observation vector vs NN input for xdpole environment.

Best generation sample

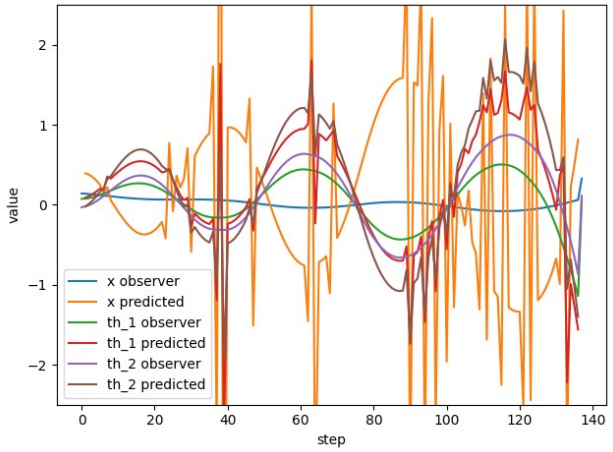


nh = 10

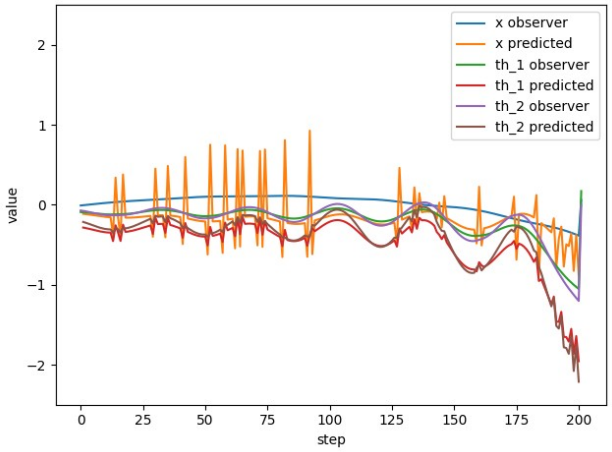


nh = 18

Best sample



nh = 10



nh = 18

Conclusions

From graphs above I may conclude that hypothesis was correct for the case of our agents. Smart alteration of locomotors observer indeed helped them learn the way to achieve stated goal. I could see that on the graphs of first variation. Although, second variation was not so fruitful, I would try to explain that behavior below.

In the graphs of the first variation I could see that the valid strategy is to alterate the position of the cart, while the angles of the poles should be left unchanged. In my opinion it is understandable behavior. If we would modify the pole angles we will only increase the complexity of original system. However, I could not state that modification of the cart position is the unite solution. Maybe we would find the stable solution with modification of angles if we will conduct enough number of an experiments.

Also, for final prove of that behavior I must thoroughly analyze the xant locomotor. I will definitely do it later.

In the second variation main problem is inconsistency of neural network. I even do not know whether such networks could be trained with any algorithm. I have not found any articles on that theme. I could see that algorithm overestimates the change of the position, and when the observation is hidden, prediction of neural network undergo a drastic changes .

I have an assumption that to train such network another term should be added to the fitness function. That term should provide the reward for minimization of difference between the predicted state and the actual observation. However that would reject the hypothesis of **(Freeman, Metz, and Ha 2019)**

My second assumption that increase of the learning time and variance also could solve that problem. The reason is that in the case of xdpole, prediction neurons used only 1/5 of all the network training time for properly adjusting their values. Also we could see the growth of the fitness in the end of xant environment (what used 4/5 of overhaul network training time) training, that means that it still have potential for grow.

Another interesting for analysis problem is my wrong results, cause they show that sharing some common information between neural agents could help to train them.

All that was done, confirms the hypothesis described in (Freeman, Metz, and Ha 2019) in case of the first variation.
To made valid conclusion concerning the second variation additional experiments must be conducted.

References

Freeman, C. Daniel, Luke Metz, and David Ha. 2019. “Learning to Predict Without Looking Ahead: World Models Without Forward Prediction,” no. 9: 1–12. http://arxiv.org/abs/1910.13038.

Mnih, Volodymyr, David Silver, and Martin Riedmiller. n.d. “Deep Q Network (Google),” 1–9.

Pagliuca, Paolo, Nicola Milano, and Stefano Nolfi. 2018. “Maximizing Adaptive Power in Neuroevolution,” 1–27.