CS294-112 Deep Reinforcement Learning HW5c
Meta-Reinforcement Learning

Louis Tilloy

Problem 1 : Context as Task ID

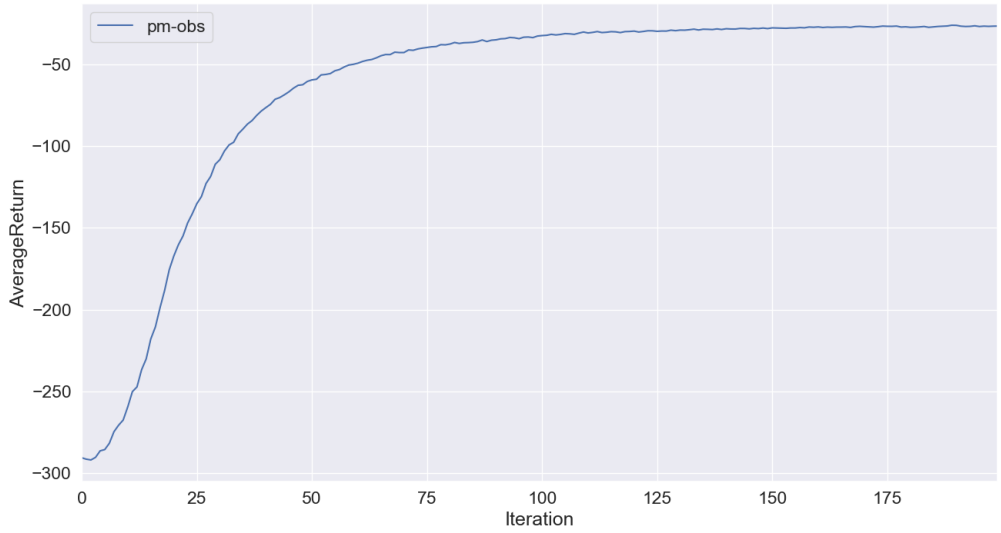

Figure 1 – AverageReturn evolution as a function of number of training iterations

Data for the graph obtained by running :

```
python train_policy.py 'pm-obs' --exp_name <experiment_name>
    --history 1 -lr 5e-5 -n 200 --num_tasks 4
```

Problem 2 : Meta-Learned Context

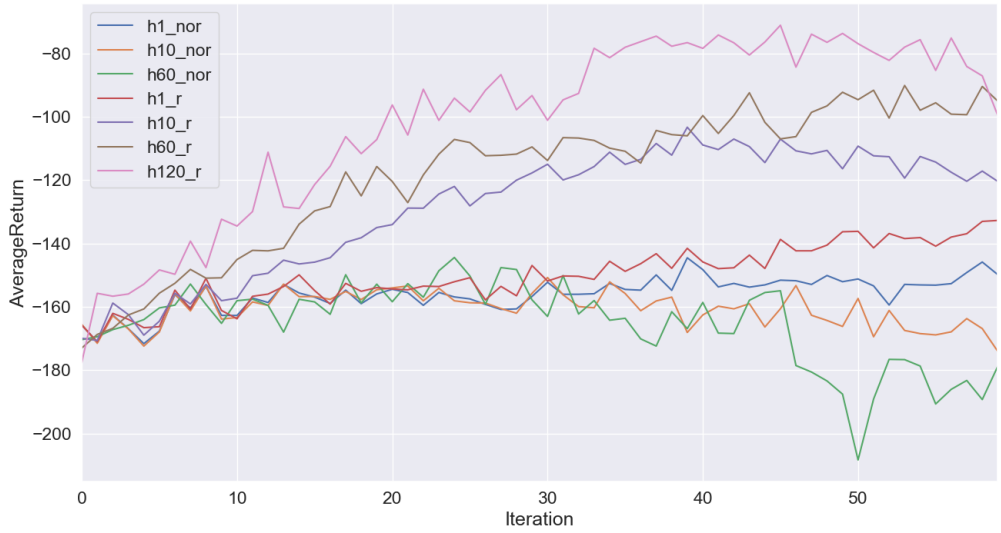

FIGURE 2 – AverageReturn evolution as a function of number of training iterations (h : history length, nor/r : multi-layer perceptron or recurrent network)

Data for the graph obtained by running :

```
python train_policy.py pm --exp_name q2_h1_nor --history 1 --discount 0.90 -lr 5e-4 -n 60
python train_policy.py pm --exp_name q2_h10_nor --history 10 --discount 0.90 -lr 5e-4 -n 60
python train_policy.py pm --exp_name q2_h60_nor --history 60 --discount 0.90 -lr 5e-4 -n 60

python train_policy.py pm --exp_name q2_h1_r --history 1 --discount 0.90 -lr 5e-4 -n 60
    --recurrent
python train_policy.py pm --exp_name q2_h10_r --history 10 --discount 0.90 -lr 5e-4 -n 60
    --recurrent
python train_policy.py pm --exp_name q2_h60_r --history 60 --discount 0.90 -lr 5e-4 -n 60
    --recurrent
python train_policy.py pm --exp_name q2_h120_r --history 120 --discount 0.90 -lr 5e-4 -n 60
    --recurrent
```

We can see that for the multi-layer perceptron, increasing the history length actually lowers the network performances. The explanation is most probably that increasing the history increases proportionally the number of weights of the network, making it harder to train (with $h = 60$, there are 60 times more weights to train than for $h = 1$).
The recurrent neural network does not suffer from this increasing number of parameters since whatever the history, the number of parameters of the GRU cell is always the same. Therefore we can see that the network is able to use the history information by deducing the task it needs to do in order to have better performance.
Overall, the recurrent neural network has better performance than the multi-layer-perceptron. As for the minimum history length, 60 seems like a good value since it achieves more than -100 AverageReturn, but there is not really a minimum value since increasing the history always increases the performance of the network.

Problem 3 : Generalization

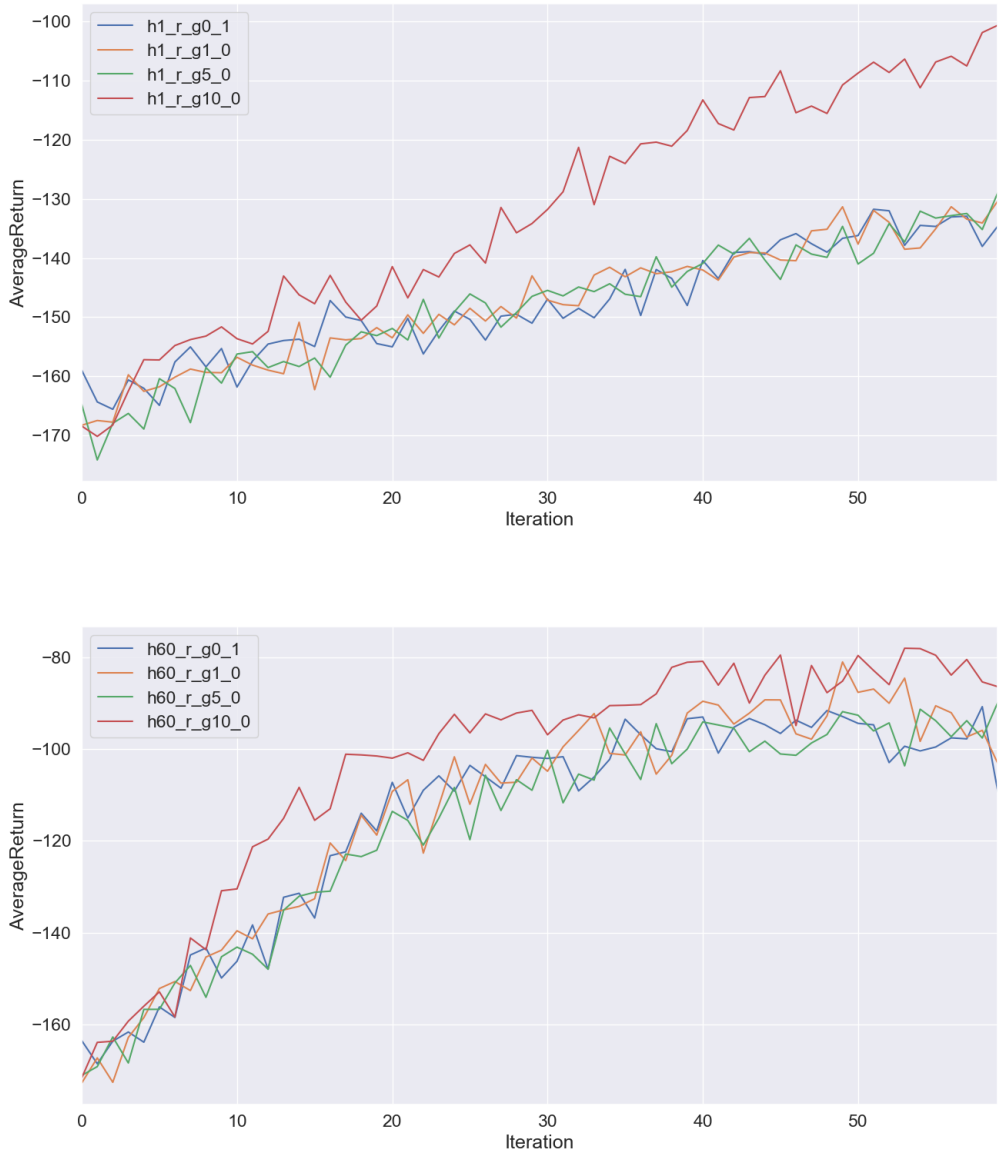

FIGURE 3 – **AverageReturn** (on training data distribution) evolution as a function of number of training iterations for recurrent network (h : history length, gx_y : inverse granularity of x.y)

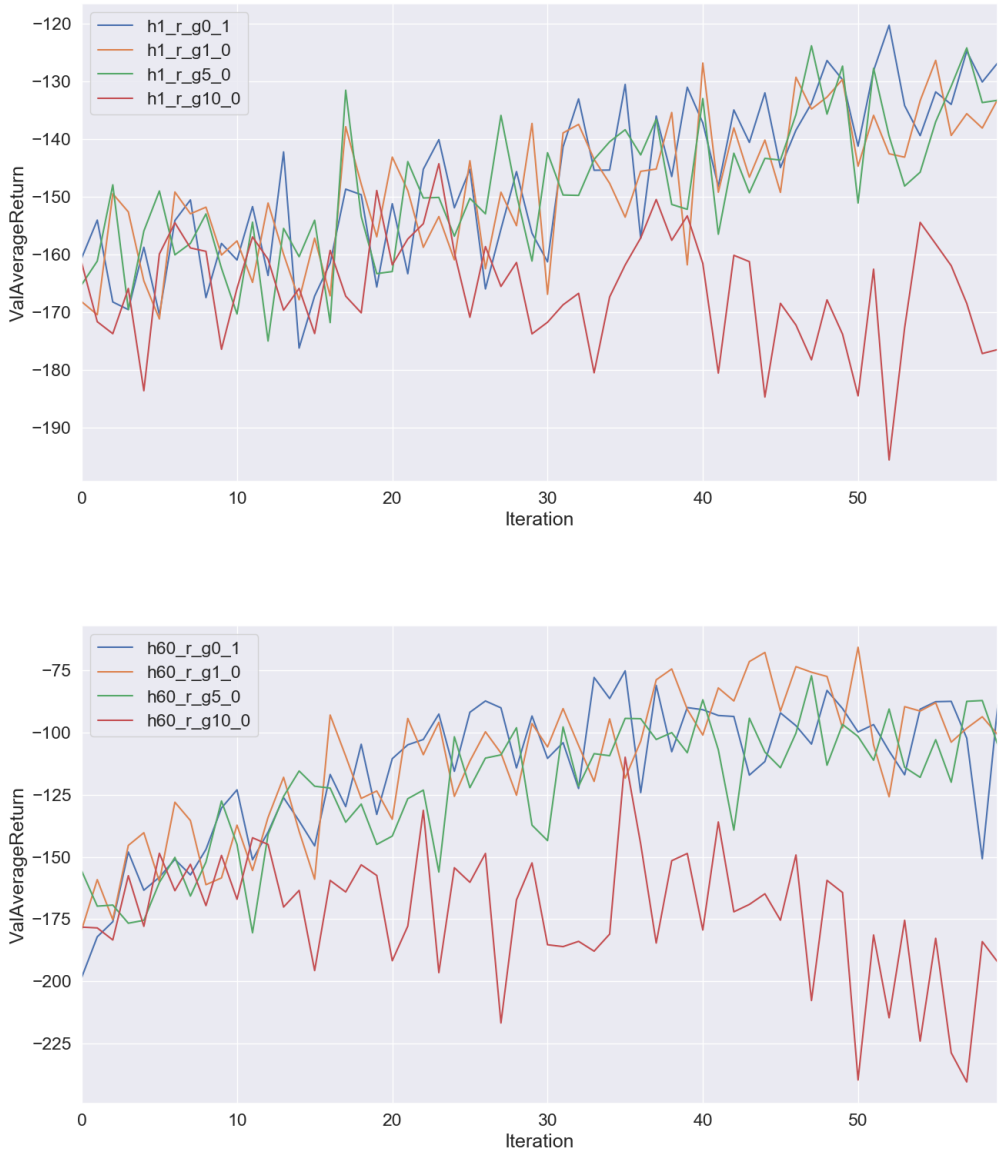

FIGURE 4 – **ValAverageReturn** (on evaluation data distribution) evolution as a function of number of training iterations for recurrent network (h : history length, gx_y : inverse granularity of x.y)

Data for the graphs obtained by running :

```
python train_policy.py pm --exp_name q3_h60_r_g0_1 --history 60 --discount 0.90 -lr 5e-4 -n 60
 --recurrent -g 0.1
python train_policy.py pm --exp_name q3_h60_r_g1_0 --history 60 --discount 0.90 -lr 5e-4 -n 60
 --recurrent -g 1
python train_policy.py pm --exp_name q3_h60_r_g5_0 --history 60 --discount 0.90 -lr 5e-4 -n 60
 --recurrent -g 5
python train_policy.py pm --exp_name q3_h60_r_g10_0 --history 60 --discount 0.90 -lr 5e-4 -n 60
 --recurrent -g 10

python train_policy.py pm --exp_name q3_h1_r_g0_1 --history 1 --discount 0.90 -lr 5e-4 -n 60
 --recurrent -g 0.1
python train_policy.py pm --exp_name q3_h1_r_g1_0 --history 1 --discount 0.90 -lr 5e-4 -n 60
 --recurrent -g 1
python train_policy.py pm --exp_name q3_h1_r_g5_0 --history 1 --discount 0.90 -lr 5e-4 -n 60
 --recurrent -g 5
python train_policy.py pm --exp_name q3_h1_r_g10_0 --history 1 --discount 0.90 -lr 5e-4 -n 60
 --recurrent -g 10
```

We can see that the less granularity (or the more inverse granularity), the easier it is for the neural network to learn how to predict the training goals, and the harder it is to learn how to predict the evaluation goals. It seems logical since it is harder to learn a more complex mapping of the space for the training, but the more this mapping is close to a uniform mapping, the easier it is for the network to generalize.
We can also see that the granularity does not influence a lot the performance of the neural network on the evaluation data up to some limit value (roughly inverse granularity > 5.0). Even with a big inverse granularity of 5.0, it manages to have an ValAverageReturn of up to -80 which is similar to the performance on the training data (see figure 2). It means that the neural network manages to generalize to goal values outside of the training distribution.

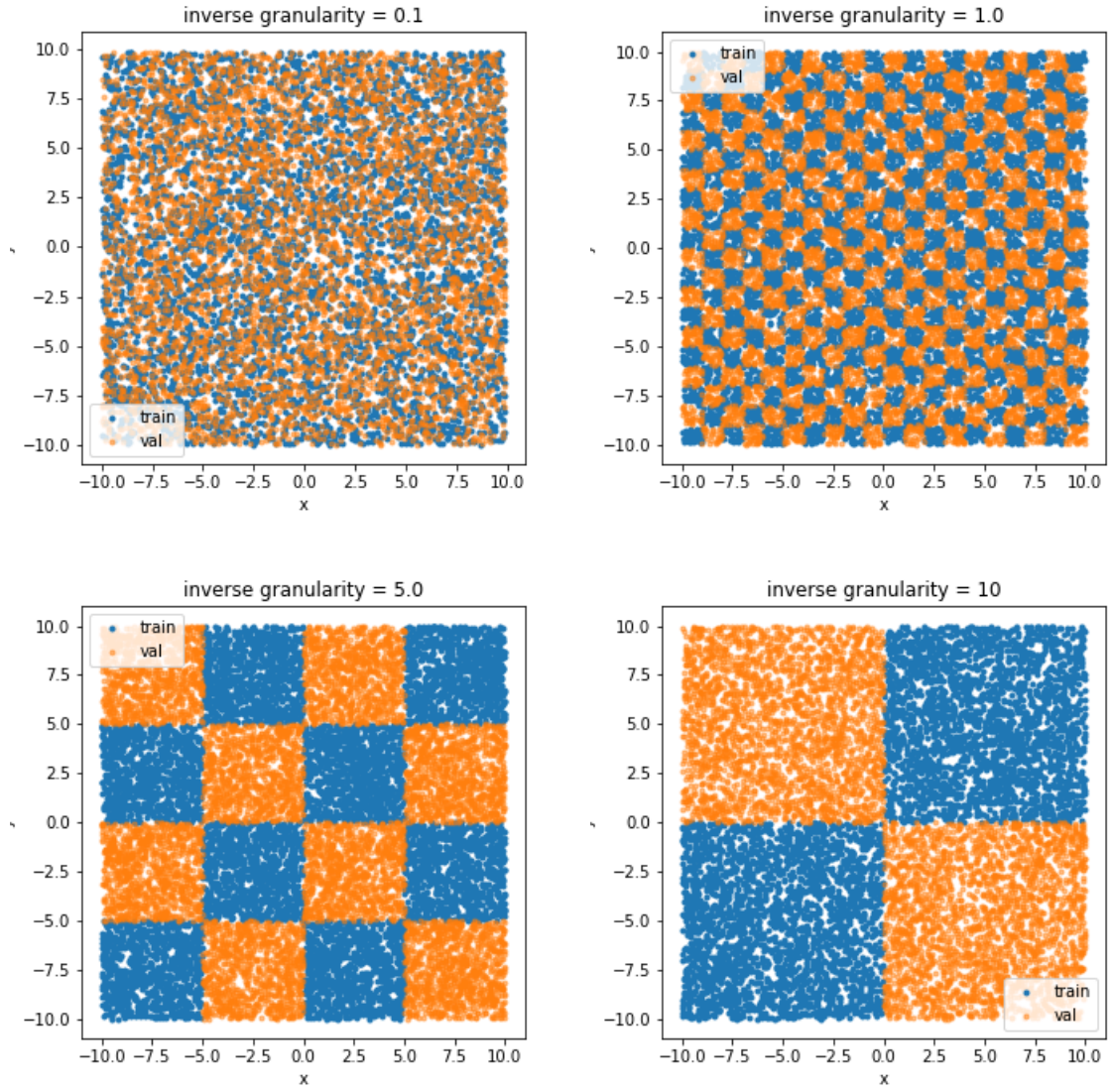

FIGURE 5 – Visualization of the goals granularity

(figures obtained by running the jupyter notebook *plot_granularity.ipynb*)