University of California Berkeley

Computer Science Department

CS 294-112

HW5c: Meta Learning DRL

Submitted by
Cristobal Pais

CS 294-112
Nov 14th
Fall 2018

HW5c: Meta Learning DRL

Cristobal Pais

Nov 14th 2018

1 Problem 1: Context as Task ID

The command used to generate the results of this section is:

```
python train_policy.py 'pm-obs' --exp_name P1 --history 1 -lr 5e-5
                        -n 200 --num_tasks 4
python plot.py data/P1 --plotName P1_NN
```

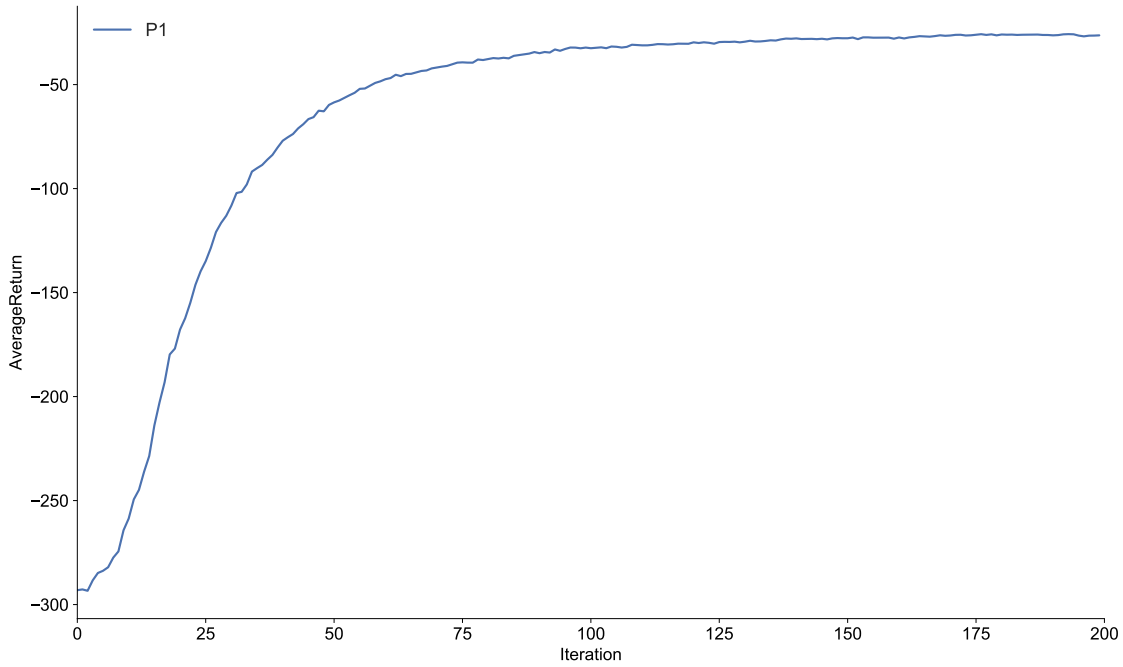


Figure 1: Average return when telling the policy which task it should perform, rather than asking it to figure it out from reward signals. As can be seen, it converges to the theoretical performance bound close to -50 after 100 iterations.

2 Problem 2: Meta-Learned Context

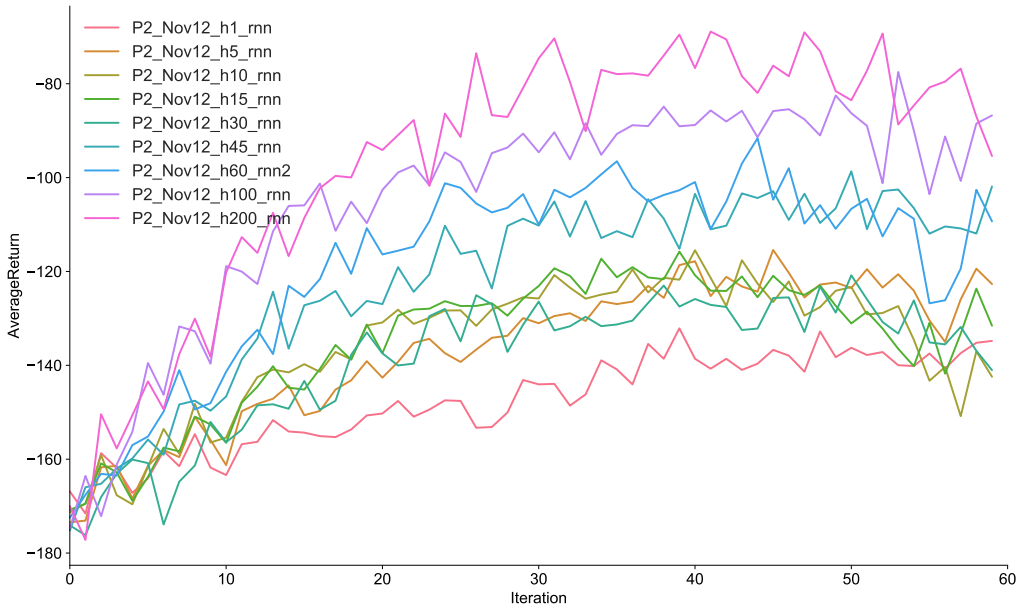


Figure 2: Recurrent Neural Network (RNN) performance for different history lengths. As expected, the longer the meta observation vector (more history) the better the performance of the agent since it is able to get more contextual information regarding the task being performed, inferring the best decisions for the task by exploiting the information of each time-step included in the replay buffer. Best performance is achieved with a history length of 200 (around -80) while the expected -100 average return level is achieved when setting it to 60 time-steps. Based on the trade-off between running time and performance, a history length of 100 seems to be optimal.

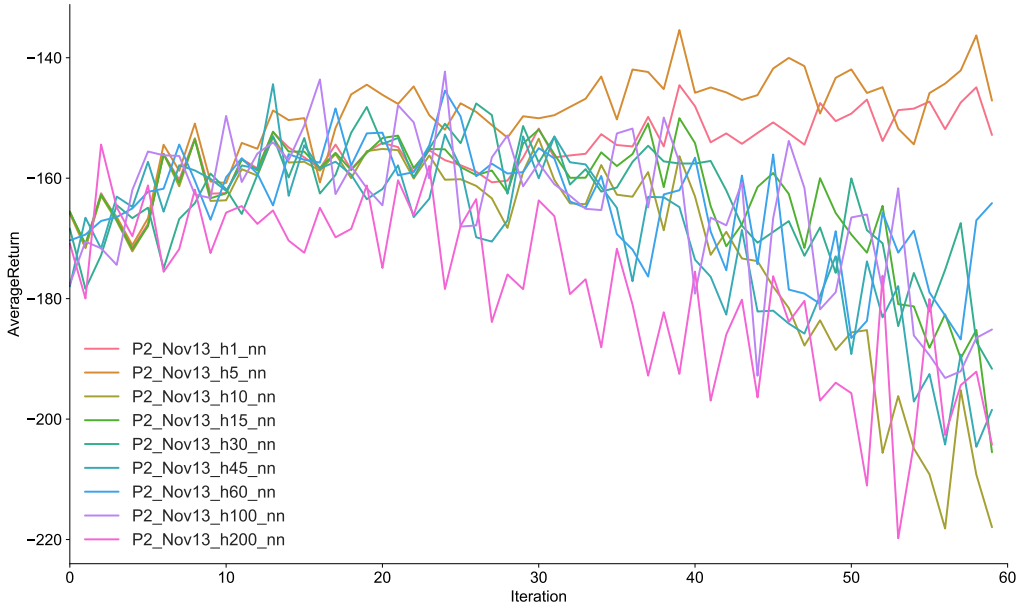


Figure 3: Feed Forward Neural Network (MLP) performance for different history length. As expected, adding more time steps to the meta observation array (and thus, the replay buffer) is not improving the performance of the agent: it is not able to understand and exploit the contextual information since no information is captured every time iteration like in the RNN implementation. Therefore, a maximum average return around -140 is obtained in the best case (history equal to 5 or 1). Performance tends to degrade with larger history, however, this significantly depends on the random seed selected during the experiment, not allowing us to indicate it as a performance pattern.

The commands used to generate the results of this section are:

```
python train_policy.py 'pm' --exp_name P2_NN_H1 --history 1
                            --discount 0.90 -lr 5e-4 -n 60
python train_policy.py 'pm' --exp_name P2_NN_H5 --history 5
                            --discount 0.90 -lr 5e-4 -n 60
python train_policy.py 'pm' --exp_name P2_NN_H10 --history 10
                            --discount 0.90 -lr 5e-4 -n 60
python train_policy.py 'pm' --exp_name P2_NN_H15 --history 15
                            --discount 0.90 -lr 5e-4 -n 60
python train_policy.py 'pm' --exp_name P2_NN_H30 --history 30
                            --discount 0.90 -lr 5e-4 -n 60
python train_policy.py 'pm' --exp_name P2_NN_H45 --history 45
                            --discount 0.90 -lr 5e-4 -n 45
python train_policy.py 'pm' --exp_name P2_NN_H60 --history 60
                            --discount 0.90 -lr 5e-4 -n 60
python train_policy.py 'pm' --exp_name P2_NN_H100 --history 100
                            --discount 0.90 -lr 5e-4 -n 60
python train_policy.py 'pm' --exp_name P2_NN_H200 --history 200
                            --discount 0.90 -lr 5e-4 -n 60
```

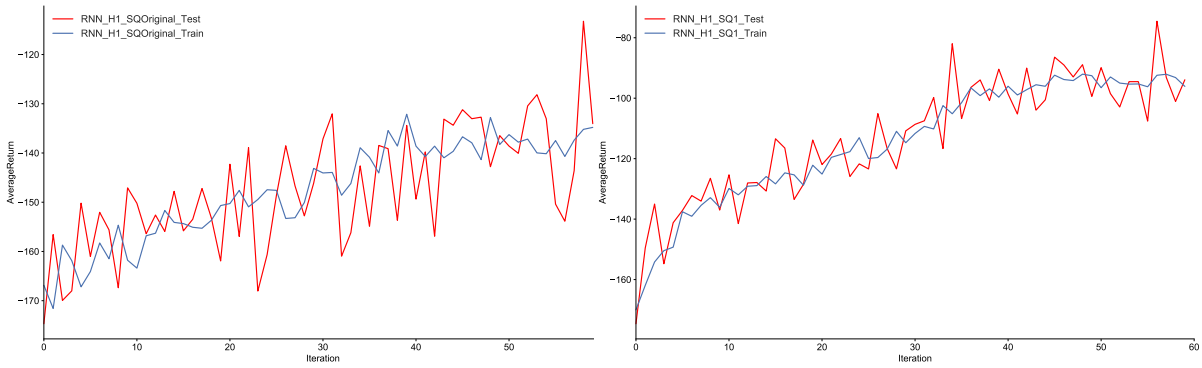
```
python train_policy.py 'pm' --exp_name P2_RNN_H1 --history 1
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
python train_policy.py 'pm' --exp_name P2_RNN_H5 --history 5
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
python train_policy.py 'pm' --exp_name P2_RNN_H10 --history 10
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
python train_policy.py 'pm' --exp_name P2_RNN_H15 --history 15
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
python train_policy.py 'pm' --exp_name P2_RNN_H30 --history 30
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
python train_policy.py 'pm' --exp_name P2_RNN_H45 --history 45
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
python train_policy.py 'pm' --exp_name P2_RNN_H60 --history 60
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
python train_policy.py 'pm' --exp_name P2_RNN_H100 --history 100
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
python train_policy.py 'pm' --exp_name P2_RNN_H200 --history 200
                            --discount 0.90 -lr 5e-4 -n 60
                            --recurrent
```

```
python plot.py data/Folder_1 ... data/Folder_n --plotName P2_RNN
```

3 Problem 3: Generalization

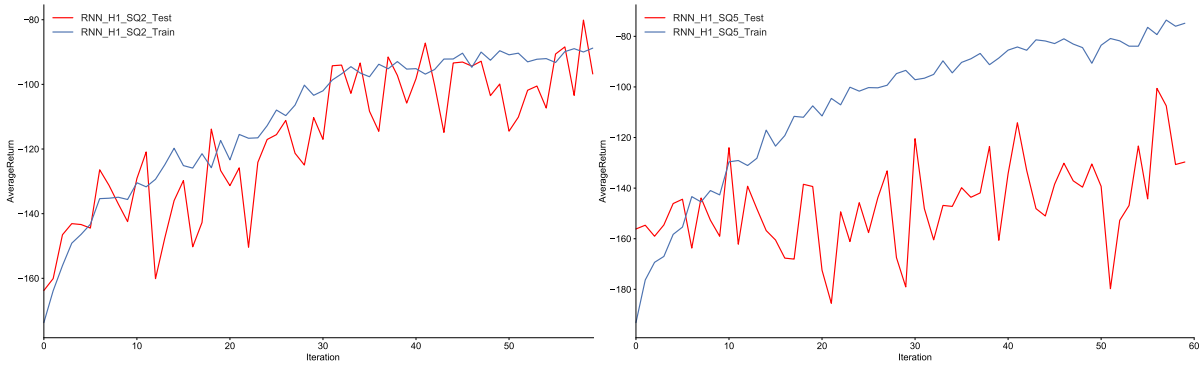
Two main experiments are conducted: history length equal to 1 or 60, varying the size of the squares inside the checkerboard from $\{1, 2, 5, 10\}$ in addition to the original implementation (original environment), using our Recurrent Neural Network from the previous question.

3.1 History length: 1



(a) RNN, history length = 1, original environment (b) RNN, history length = 1, square size = 1

Figure 4: When comparing the generalization performance of the original environment and the modified one using a checkerboard granularity of 1 (square size equals to 1) we cannot see significant differences, besides a slightly better performance when using square size = 1. However, we expect worse performance when increasing the size of the square as both the training and testing goals will vary significantly.



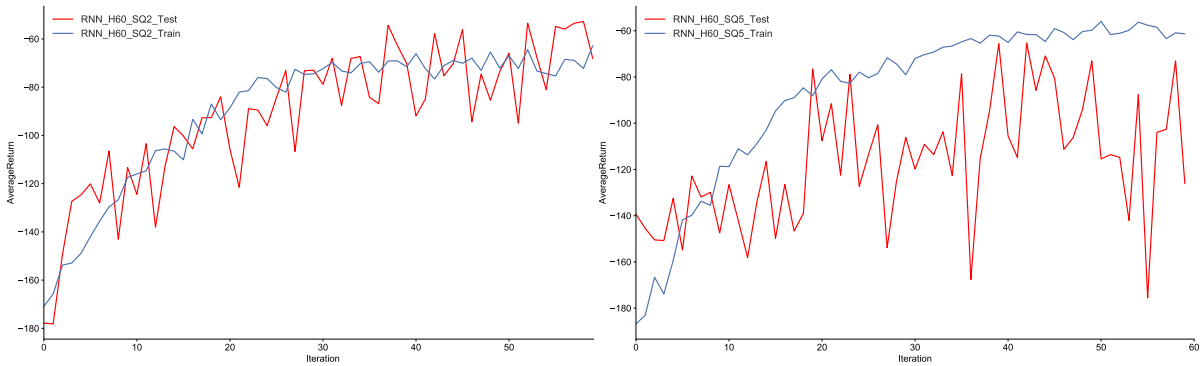
(a) RNN, history length = 1, square size = 2 (b) RNN, history length = 1, square size = 5

Figure 5: When increasing the granularity, up to a square size equal to 5, we can see how the performance on the testing set is degraded and thus, the generalization performance of our agent is getting worse, as expected.

3.2 History length: 60

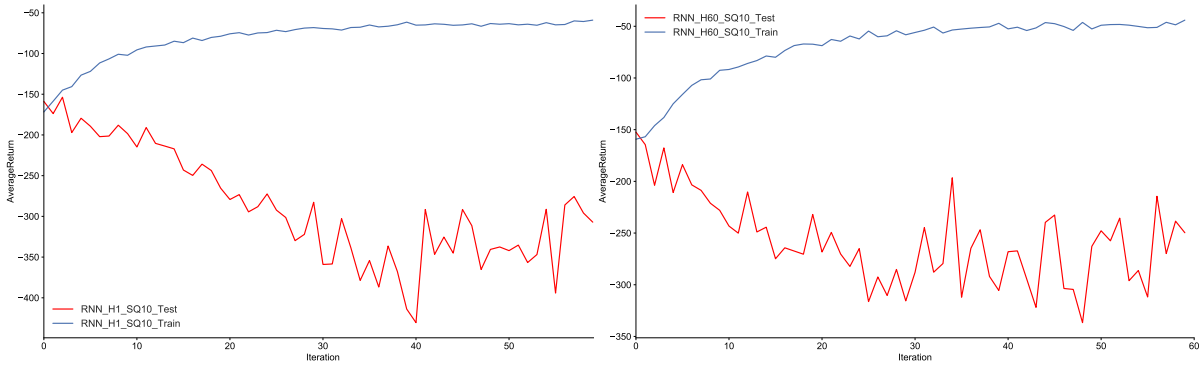
(a) RNN, history length = 60, original environment (b) RNN, history length = 60, Square size = 1

Figure 6: Similar patterns can be seen when using 60 time-steps for the meta observations (history), just modifying the final average performance of the agent (better than with history length = 1).



(a) RNN, history length = 60, square size = 2 (b) RNN, history length = 60, square size = 5

Figure 7: Increasing the size of the squares inside the checkerboard tends to decrease the performance of the agent over the testing set, as the generalization is getting worse due to the granularity and differences between both training and testing sets, as can be seen when we increase the square size up to 5 units.



(a) RNN, history length = 1, square size = 10 (b) RNN, history length = 60, square size = 10

Figure 8: In the extreme case where squares are of size equal to 10 (4 squares checkerboard), we can clearly see how the generalization performance is significantly degraded in this setting no matter the history length: Training set performance tends to be very smooth and increasing while the testing set is not able to achieve good performance, reaching very low average return levels (below -300) on this set. Thus, we are able to understand the impact on our policy performance as the training and testing distributions become more different.

The commands used to generate the results and figures of this section are (including a modified version of the plot.py script named plot_P3.py that includes the testing vs training comparison):

```
python train_policy.py 'pm' --exp_name P3_H1_SQ1 --history 1
                            --discount 0.90 -lr 5e-4 -n 60 --sqs 1
                            --recurrent
python train_policy.py 'pm' --exp_name P3_H1_SQ2 --history 1
                            --discount 0.90 -lr 5e-4 -n 60 --sqs 2
                            --recurrent
python train_policy.py 'pm' --exp_name P3_H1_SQ5 --history 1
                            --discount 0.90 -lr 5e-4 -n 60 --sqs 5
                            --recurrent
python train_policy.py 'pm' --exp_name P3_H1_SQ10 --history 1
                            --discount 0.90 -lr 5e-4 -n 60 --sqs 10
                            --recurrent
```

```
python train_policy.py 'pm' --exp_name P3_H60_SQ1 --history 60
                            --discount 0.90 -lr 5e-4 -n 60 --sqs 1
                            --recurrent
python train_policy.py 'pm' --exp_name P3_H60_SQ2 --history 60
                            --discount 0.90 -lr 5e-4 -n 60 --sqs 2
                            --recurrent
python train_policy.py 'pm' --exp_name P3_H60_SQ5 --history 60
                            --discount 0.90 -lr 5e-4 -n 60 --sqs 5
                            --recurrent
python train_policy.py 'pm' --exp_name P3_H60_SQ10 --history 60
                            --discount 0.90 -lr 5e-4 -n 60 --sqs 10
                            --recurrent
```

```
python plot_P3.py data/P3_H<h>_SQ<sq> --plotName P3_RNN<h>_SQ<sq>
                  --label RNN_H<h>_SQ<sq>
```