# RL projects

Snigdha Dagar, Frederic Alexandre

November 2020

## 1 Task Protocol

### 1.1 The Task

The task is a spatial alternation task, in the environment as shown in Figure 1. The agent begins in the bottom of the central hallways (square marked S) and proceeds up to the choice point. On the first (sample) trial, the agent can either turn left or right and receives a positive reward at the end of the arm (squares marked G). The agent then continues along the return arm and back to the starting point (where it is prevented from entering the side hallway by a barrier). Following the first trial, the agent receives a positive reward if it chooses the opposite direction as on the previous trial, otherwise it receives a negative reward. This task is non-markov as the correct response depends on the agent's choice in the previous trial.

### 1.2 Reinforcement Learning Model

A tabular, actor critic temporal difference learning architecture with $\epsilon$ greedy exploration was used.
Working memory : Used to maintain a representation of the stimulus online. An additional action was introduced - which sets the working memory state to the current sensory location state, until it is overwritten when the next memory action is taken. The total number of possible WM states is one more than the number of location states (one extra for empty memory at the start of the episode). To maintain a working memory, a factored state space representation was used : $S : S_L x S_{WM}$ with each double of the form (grid world location, working memory contents) having its own set of action values.
For more details, refer to [1]

### 1.3 Environment Abstraction

Each of the blocks in the grid (apart from the black blocks) 1 was considered a location state (making a total of 12 states). Along with the memory states, this makes a factored state space of (12 x 13). In each of the location states, the agent was free to choose between 5 actions : up, down, right, left or update.

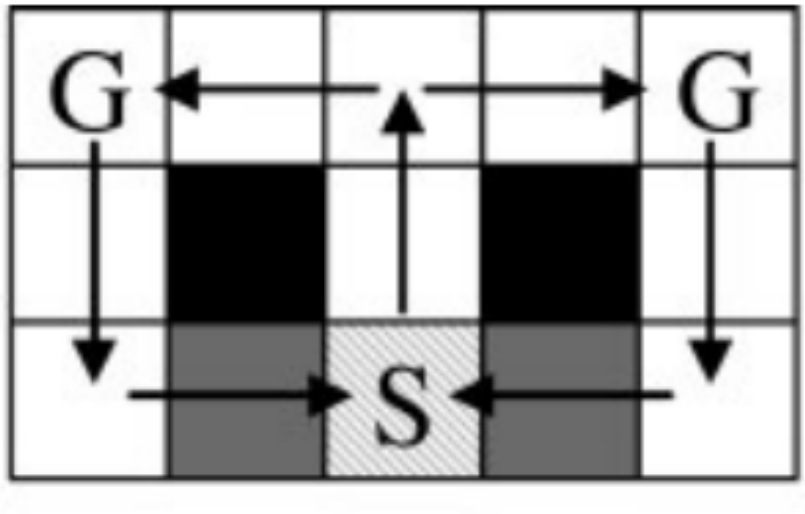


Figure 1: Task environment

The first 4 motor actions allow the agent to move to a new location state in the maze, while the update action updated the contents of the working memory to that of the current location state. Initially, before the first trial, the location state was S and the memory was empty. At all location states, except the choice point, only one action was considered 'legal' (ie moving forward). Any other action, that is attempting to move into a barrier or moving back to the previous location, or storing a state already in memory, incurred a negative reward of -1. Legal moves got a reward feedback of -0.05. At the choice point, moving left or right were both legal actions. Following the first trial, reaching the states G, got either a positive reward of +9.5 for a correct choice or a negative reward of -6 for an incorrect choice.

### 1.4 Guidelines

The code for the task can be found at Reinforcement Learning. There are two versions of the code : the simple Actor-Critic model and the A/C model with working memory. For any queries, you can write to snigdha.dagar@inria.fr

- Plot the performance of the agent as the number of correct responses over the number of trials in each block as in figure 2. Each block is of 2000 steps. Each action is considered one time step. A new trial is considered when the agent renters the starting state.
- Report the performance/learning curve as an average over 50 runs.

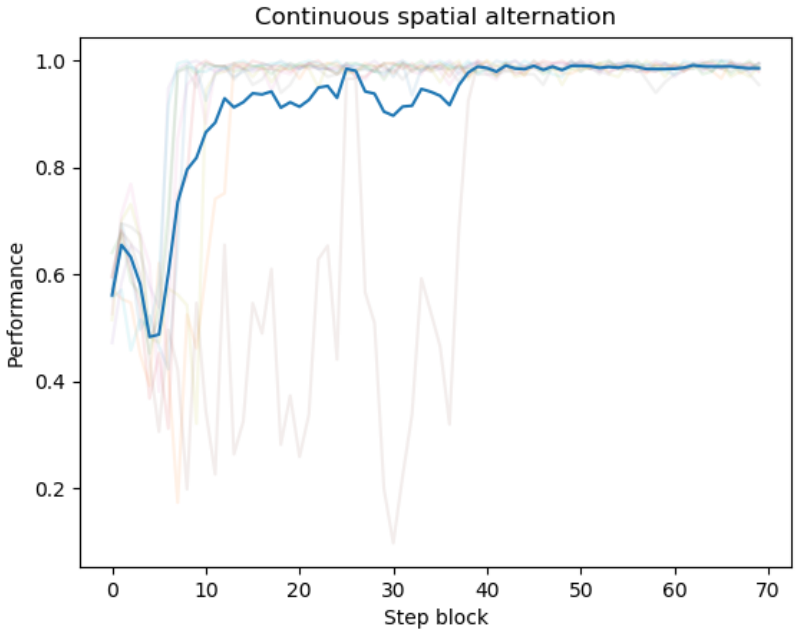


Figure 2: Average performance of the algorithm over 10 runs. Performance in each block of 2000 steps is the number of correct responses divided by the total number of responses.

- At the end of the project, create and upload a zipped folder with the following two things :
  1. The modified code with comments about your additions (analysis and plots)
  2. A pdf report with all the figures generated, with captions and explanations.

# 2 Project Proposals

## 2.1 Eligibility Traces in the Critic

Eligibility traces are used to speed up the agent's learning of the task. With eligibility traces, multiple state and action values are updated on each step with the update scaled proportional to the amount of time because the agent last visited the state and took the action. Now receipt of a reward while entering the current state affects values not only at the previous state but also at the most recently visited states.

Implement eligibility traces in the current algorithm and report the difference in the speed of learning of the agent. How the strength of a trace decays over time is controlled by a decay parameter $0 \leq \lambda \leq 1$, where $\lambda = 0$ yields decay to 0 after a single time step and $\lambda = 1$ leads to no decay. Vary the decay parameter over this range in steps of 0.01 and plot the performance to see the parameter's effect on learning.

## 2.2 Actor Policies

The current algorithm uses an $\epsilon$ greedy exploration as the actor policy. There are other policies such as the Softmax policy or Thompson sampling. Implement these policies for the actor and report the comparison in performance on using these different policies.

## 2.3 Parameter Exploration

The current algorithm has three free parameters - the exploration rate, the temporal discounting factor and the learning rate. Vary the exploration rate ($\epsilon$) and the learning rate ($\alpha$) over the specified range and report the difference in the speed of learning (convergence time to optimal performance).

## 2.4 Reward Values

A small negative reward is used even for legal actions to speed up the agent's learning of the task. Some algorithms often also use no reward feedback until the goal state, which could be linked with a positive or negative reward. Currently, there also exists the same cost for taking a motor (external) or memory (internal) action. Vary the reward values associated with these actions and states -
1) Negative reward for illegal actions that result in no state transition, but no feedback for any legal actions until the agent lands in a goal state.
2) Different costs for taking external or internal actions.
Analyze the reasons for using the current set of reward values. Report the results as learning curves and action value changes at the choice point, and in the returning arms.

## 2.5 Strategy Evaluation

The algorithm can have multiple solutions to the task. In a "remember both" strategy, the algorithm learns to update and maintain memory of the choice point observations, whether left or right. The agent learns to take a right turn when a "left turn" (previous) observation is present in memory and to take a left turn when a "right turn" (previous) observation is present in memory. By contrast a "remember one" strategy only remembers the previous observation for either a left or right turn, remaining empty in the other case.
Find the relative frequencies of different kinds of strategy by running the model 100 times and classifying the resultant strategy as either "remember both" or

"remember one" on the basis of the proportion of final-block trials in which the direction of the previous turn was in memory at the choice point. A threshold can be used : if the proportion of trials with "left turn" in memory at the choice point was above two-thirds, this counts as a "remember left" strategy, and vice versa; if proportions for both "left turn" and "right turn" were above threshold, this can be counted as "remember both"; finally, if the threshold is not reached for either "left turn" or "right turn" trials, the classification would be "other." Also report the changes in action values of the agent for the algorithm. Refer to [2], figures 5 and 6 for the kind of visualizations expected.

# References


[1] Zilli EA, Hasselmo ME. Modeling the role of working memory and episodic memory in behavioral tasks. Hippocampus. 2008;18(2):193-209.

[2] Lloyd, Kevin, et al. "Learning to use working memory: a reinforcement learning gating model of rule acquisition in rats." Frontiers in computational neuroscience 6 (2012): 87.