Report

Models of Higher Brain Functions

Programming Project

Reinforcement learning model of spatial navigation

Authors:
Sören Becker, Jan Bölts

Supervisor:
David Higgins

August 8, 2016

1 Introduction

This report describes a computational approach of learning an optimal strategy for a navigation task in a previously unknown environment. This kind of learning objective is a classical problem in the domain of reinforcement learning.

Generally, in reinforcement learning an agent improves its performance on a given task by iteratively applying different actions, observing their consequences and learning from this experience. In the beginning of the learning process, the agent is completely ignorant to the task and has to explore its options of interaction with the environment in order to reach a predefined goal. Having reached the goal, a positive reinforcer is disposed, which indicates that the sequence of chosen actions led to the solution. The agent can use the reinforcer to update its preferences on action choices in order to improve its performance in the future.

Since the agent is ignorant to the task at the beginning, meaning that no prior well-grounded preferences exist, it needs to perform random actions until it reaches the goal. On the second attempt to the task, the agent could in principle repeat the previous sequence of actions as this already led to the goal on the previous attempt. However, there is in general no justification to believe that the previous sequence of actions is optimal in the sense that it represents the shortest or least costly solution. The agent can therefore not simply exploit its experience. Hence, a major challenge in reinforcement learning, which is known as the exploration-exploitation-dilemma is to find a good trade-off between active exploration of possible actions and exploitation of already gathered experience. One way of dealing with this dilemma is to let the agent act according to some kind of policy which allows both random exploration of actions and action choices based upon experience. The ratio of random choices can then be gradually lowered with the increasing number of repetitions of the task to increase the reliance on experience. This broad concept of reinforcement learning is inspired by the introductions given in [1] and [2] and describes the general learning framework that this report is concerned with.

In particular, in the experiment investigated in this report, the environment consists of a T-shaped maze and the agent's task is similar to the stick-throwing game often played with dogs. Specifically, the agent starts at the lower end of the maze and has to learn by interacting with the environment to first go to the pick-up area located at the right end of the horizontal

arm of the maze before going to the target area located at the left end of the horizontal arm. Difficulties of this task include that the agent has to learn the shape of the maze and that it needs to go to the pickup area before going to the target area as only then a reward is disposed. We investigate the agent's learning ability under a number of different parameter settings which effictively impact the number of possible action choices, the trade-off between exploration and exploitation, and the effect of memory-decay of an action's consequences.

The report is organized in the following way: an informal description of reinforcement learning and the investigated task is given in this section, a detailed, formal description of the simulated experiment and the employed methods is provided in section 2. In section 3 we describe the results of different parameter settings on the agent's learning ability which are discussed and explained in section 4.

2 Methods

This section provides a formal description of the simulated environment and task intended to be solved, the agent's internal representations, and lastly the applied learning algorithm.

2.1 Environment & Task:

The environment for the simulation is a T-shaped maze. The horizontal arm of the maze has a length of 110 cm and the vertical arm has a length of 50 cm while both have a width of 10 cm. The pickup and reward area cover the endmost 20 cm on both sides of the horizontal arm respectively. The geometry of the environment is depicted in Figure 1. In each simulation, the agent starts at the lower end of the maze and has to move into the pickup area before proceeding to the reward area to successfully finish one trial. If the agent attempts to step out of the boundaries of the maze, it receives a negative reward of -1 but remains at its current position, i.e., the illegal step out of the maze is not carried out. Furthermore, the agent receives a positive reward of +20 if it steps into the reward area after visiting the pickup area which automatically terminates a trial as this represents a completion of the task. No other rewards are disposed and in particular no reward is disposed at the target area if the pickup area has not been visited yet.

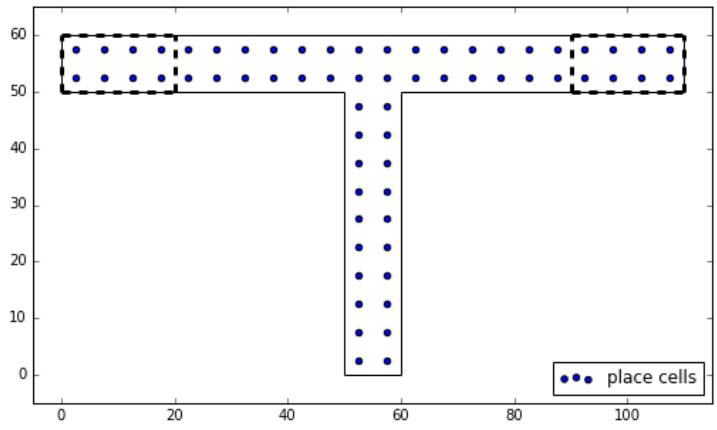

Figure 1: The geometry of the environment. The wall of the T-shaped maze are indicated by black lines, the target areas by dotted lines and the place cell centers by blue dots.

2.2 Agent

The agent's representation of the environment consists of two populations each containing n = 64 neurons. Specifically, the maze is divided by a regular grid into 64 fields of equal size and each neuron's activity is modeled as a location-dependent radial basis function that is placed at the center of one of these fields. One of the two populations is only active while the agent has not yet visited the pickup area whereas the other population is only active while the agent has already visited the pickup area. Formally, the activation of a neuron is given by

$$r_{j,\beta}(s) = \delta_{\alpha\beta} \exp(-\frac{(x_j - x)^2 + (y_j - y)^2}{2\sigma^2}) \tag{1}$$

where $j = 1, ..., 64$ indexes a neuron in population $\beta = \{0, 1\}$. Variable $\alpha = \{0, 1\}$ expresses whether the pickup area has been visited and $\delta_{\alpha,\beta}$ denotes Kronecker's delta, which ensures that only one of the two population is active. The placement of the radial basis function on the maze is given by (x_j, y_j) and its width is set to $\sigma = 5$ throughout all simulations. Each radial basis function is thus effectively a function of the triplet $s = (x, y, \alpha)$, i.e., the position and a boolean variable indicating a visit of the pickup area. This arangement of a spatial representation resembles place cells found in the hippocampus of rats and we therefore adopt the term place cells henceforth.

The activations of the place cells are linearly combined to obtain a set of output neurons $Q_1, ..., Q_m$ such that

$$Q_a(s) = \sum_{j,\beta} w_{aj\alpha} r_{j\beta}(s) \tag{2}$$

The output neurons are indexed by $a = 1, ..., m$, where m is the number of possible movements of the agent. The activation of the output neurons represent the action preferences for each state $s = (x, y, \alpha)$ and are optimized by adapting the weights $w_{aj\alpha}$ according to the learning algorithm. The agent moves through the maze by selecting a direction θ_i from a set $\{\theta_1 = 0, ..., \theta_m = \frac{m-1}{0.5 \cdot m} \cdot \pi\}$ and updating its position according to

$$x \leftarrow x + s \cdot \sin(\theta_i) \tag{3}$$

$$y \leftarrow y + s \cdot \cos(\theta_i) \tag{4}$$

For each step of the agent, the stepsize s is drawn as a random sample from a normal distribution with $\mu = 3$ and $\sigma = 1.5$. We investigated the agent's behavior for $m = \{4, 8, 16\}$ directions which always covered 2π with a regular angular spacing.

2.3 Learning algorithm

The weights $w_{aj\alpha}$ were optimized accoring to the SARSA algorithm in conjunction with an ϵ-greedy policy for direction choices. The ϵ-greedy policy allows the agent at each step with probability ϵ to select a random direction and forces it with probability $1-\epsilon$ to select the direction whose corresponding output neuron exhibits the highest activity:

$$\theta = \begin{cases} \mathcal{U}\{0, \theta_m\} & \text{with prob} = \epsilon \\ \theta_{a^*} \text{ for } a^* = \arg\max_a Q_a(s) & \text{with prob} = 1 - \epsilon \end{cases} \tag{5}$$

Policy parameter ϵ captures the trade-off between exploration and exploitation and decays exponentially at each step according to

$$\epsilon_{t+1} \leftarrow \epsilon_t \cdot 0.1^{\frac{1}{0.7 \cdot T}} \tag{6}$$

where T is the number of simulated trials. When ϵ became lower that 0.1, the decay was stopped and ϵ remained constant for the remaining trials.

Based on the activity of the output neurons and the received rewards the SARSA algorithm iteratively adjusts the weights $w_{aj\alpha}$ at each time step t of a trial by adding

$$\Delta w_{aj\beta} = \eta \delta_t e_{aj}. \tag{7}$$

Here, $\eta = 0.1$ is the learning rate, δ_t is a temporal difference factor. It is calculated from the reward r_t received for the action to be performed, the activity of the output neurons corresponding to the action at the next time step a_{t+1} and the previous action a_t:

$$\delta_t = r_t + \gamma Q_{a_{t+1}}(s_{t+1}) - Q_{a_t}(s_t). \tag{8}$$

Furthermore, factor e_{aj} of equation 7 represents an eligibility trace that can be thought of as the agent's memory of an action's consequences. The memory trace is also updated at each time step t so that old memories decay but the experience of the current action's consequence is strengthend. This is formally expressed by the update rule

$$e_{aj} \leftarrow \gamma \lambda e_{aj} + r_j(s) \delta_{a,a_t}. \tag{9}$$

In equations 8 and 9, $\gamma = 0.95$ for all simulations. For parameter λ in equation 9 the impact on learning for different values in the interval (0,1) was investigated. The full algorithm is laid out below.

3 Results

Figure 2 shows the learning curve of a simulation of 10 agents (runs) for 400 trials with escape latency averaged over all animals. A learning effect can be observed already after few trials. After about 100 trials the agent seems to have found the optimal path, i.e., learning has converged. It then needs about 60 steps to obtain the reward at the target.

A visualization of the animals behavior is shown in Figure 3. We plotted a vector field showing the directions of preferred actions inferred from the maximal Q-value for the center of every place field. The two subplots refer to the two subtask of finding the pickup area and subsequently finding the target area. For the first subtask, i.e., on the way to the pickup area, almost all preferred actions point towards to pickup area. Once it is reached, the

Algorithm 1 SARSA with function approximation

Repeat for each trial:
- $\mathbf{e} \leftarrow 0$ *Initialize eligibility trace*
- $\mathbf{w} \leftarrow 0$ *Initialize weights*
- $S' \leftarrow (x = 0, y = 0, \alpha = 0)$ *Initialize state*
- $Q \leftarrow 0$ *Initialize output neurons*
- $\epsilon_t \leftarrow 1$ *Initialize* ϵ
- $A \leftarrow a$ *Initialize action according to eq. 5*
- Repeat until S' is a terminal state:
  - $S' \leftarrow S$ *Apply A via eq. 3*
  - $A' \leftarrow A$ *Store last action*
  - $A \leftarrow a$ *Choose next action*
  - $R \leftarrow r$ *Receive reward*
  - $\delta_t \leftarrow R + \gamma \max_a Q_a(S) - Q_{A'}(S')$
  - $w \leftarrow w + \eta \delta_t e$ *Update weights*
  - $e \leftarrow \gamma \lambda e + r(s) \delta_{a,A}$ *Update eligibility trace*
  - $Q = w^T r(s)$ *Compute activity of output neurons*
  - $\epsilon = \epsilon_t \cdot 0.1^{\frac{1}{0.7 \cdot T}}$ *Let* ϵ *decay*

preference changes and the majority of vectors points towards the target area.

Figure 7 shows the development of the navigation maps over trials. The rows of subplots corresponds to the navigation map after 1 trial, 50 trials and 100 trials, respectively. One can observe a goal-directed change already after 1 trial, e.g., on the way to the target. However, clearly goal-directed behavior for all locations develops only after 100 trials.

The decay of memories seems to be a crucial point in the learning process of the agent. Figure 4 shows the learning curves for different values of the memory decay parameter λ. Note the different scales on the y-axis and the number of trials on the x-axis after which learning converges. Interestingly, learning seems to be most effective for a moderate decay around $\lambda = 0.5$. For other values learning occurs later ($\lambda = 0.2$) or not at all ($\lambda = 0.9$).

In order to copy with the exploration/exploitation dilemma it is useful to use a time-varying exploration/exploitation parameter ϵ. Figure 5 shows the time course of the ϵ parameter. It is close to 1 in the beginning and then decays exponentially to 0.1 within 70% of the given number of trials.

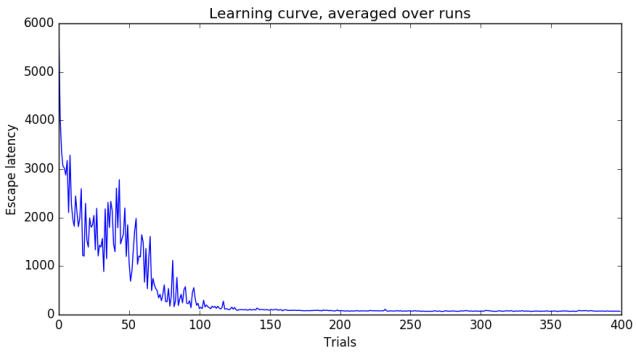

Figure 2: Learning curve averaged over 10 runs. 400 trials, 4 actions, $\gamma = 0.95$, $\lambda = 0.4$, ϵ decaying exponentially to 0.1.

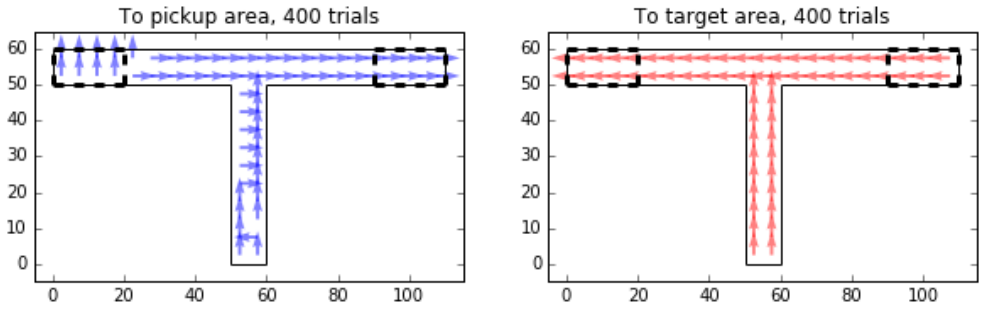

Figure 3: Prefered directions after learning.

Finally, the effect of varying number of possible actions on the learning process was investigated. Figure 8 shows the learning curves for different numbers of possible actions. The more actions the agent can choose from, the longer it needs to learn the task. Interestingly, the escape latency decreases first and then increases again for both, the case 8 and the case of 16 possible actions.

4 Discussion

This section will discuss and explain the results that were presented in the previous sections. It will comment on the question that were given in the project description.

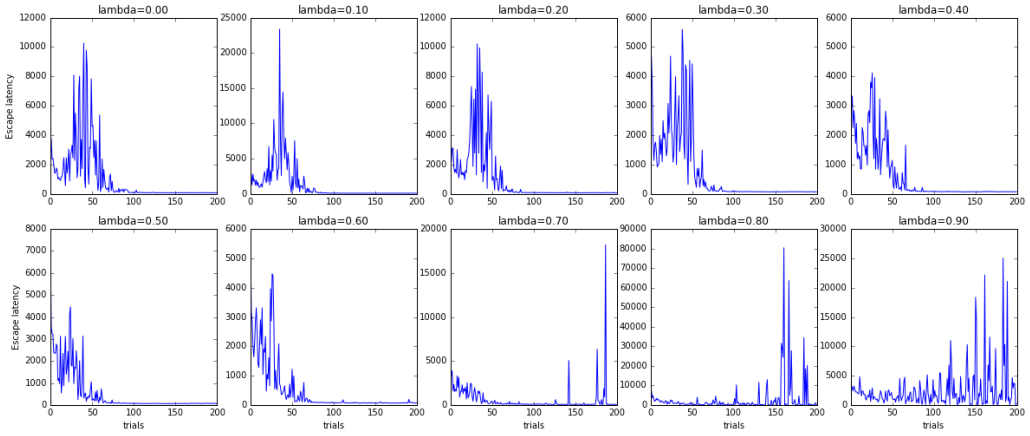

Figure 4: Learning curves for difference values of memory decay parameter λ. 200 trials, averaged over 5 runs, $\gamma = 0.95$.

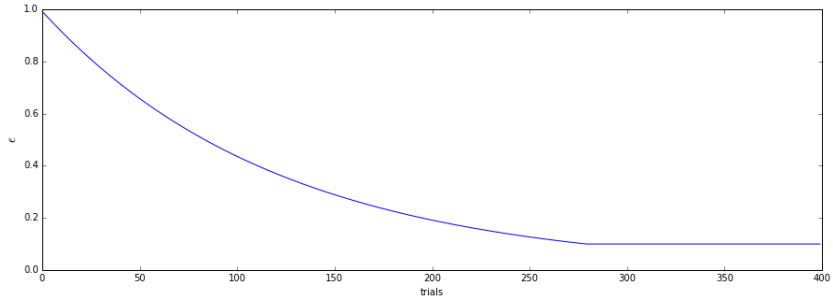

Figure 5: The time course of the ϵ parameter over 400 trials.

The initial simulation consisted in 10 runs with 400 trials each and calculated the escape latencies over trials, averaged over all runs. This was in order to quantify how long the takes the agent learn the task. We state that the agent has learned the task if there is no more, or very little, variation in the escape latencies and if the escape latency is close to the theoretical optimum. From the results given above we can see that this is the case after about 100 trials. However, this result depends on how fast the agent switches from exploration to exploitation, i.e., if we were to reduce the number of trials and increase the rate with which ϵ decays to 0.1, the agent might be able to learn even faster. This comes with the risk that it might not find the optimal path because it does not explore long enough. Figure 6 shows the learning curves after different number of trials, averaged over 5 runs. Only

70% of the given number of trials is used for learning, i.e., after that $\epsilon = 0.1$. One can observe that under these circumstances, the agent is able to learn the task within 40 trials.

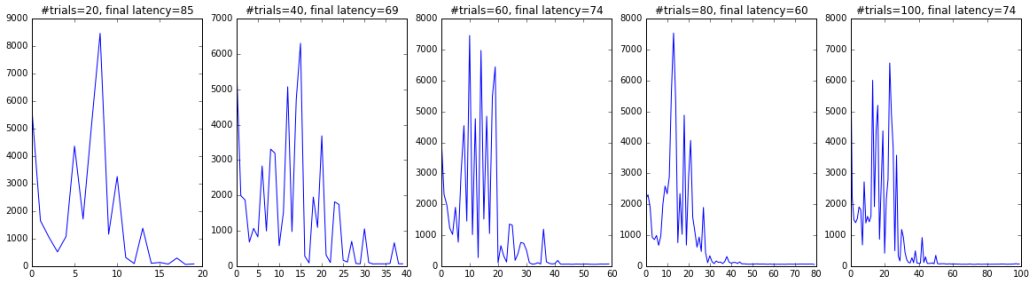

Figure 6: Learning curves for different numbers of trials. Averaged over 5 runs, $\lambda = 0.5$, 70% of trials for learning.

We presented a visualization of the agent's behavior by plotting a vector field of preferred movement direction for every place cell center position in Figure 3 and 7. The results show that for the navigation map of the subtask of finding the way to the target, many arrows point toward the target area in the upper left arm already after the first trial. For the navigation map of the subtask of finding the pickup area this is not the case. This may be due to the fact the information of the reward needs more trials to spread to earlier stages of the task. Consequently, after 50 trials, both navigation maps show clearly goal directed preferred movement directions, i.e., for both navigation maps the preferred movement direction in the vertical arm is upwards, then, before the pickup area is reached (blue arrows) the preferred movement direction in the horizontal arm is to the right and after the pickup area was reached it changes to the left (red arrows).

As described in the methods, we introduced an exponential decay over trials on the epsilon parameter (Equation 6). A higher value for epsilon leads to a larger ratio of random choices to choices based on previous experience. This parameter therefore effectively controls the trade-off between exploration and exploitation. Letting it decay from a high value at the beginning of the learning process to a low value therefore expresses the idea of letting the agent explore while it lacks experience and letting it make use of experience once it has been established. We would like to note that this does not solve the exploration-exploitation-dilemma but merely shifts it to finding the right adaptation function for parameter epsilon. In the simulations

presented here, we chose an exponential decay which is depicted in figure 5.

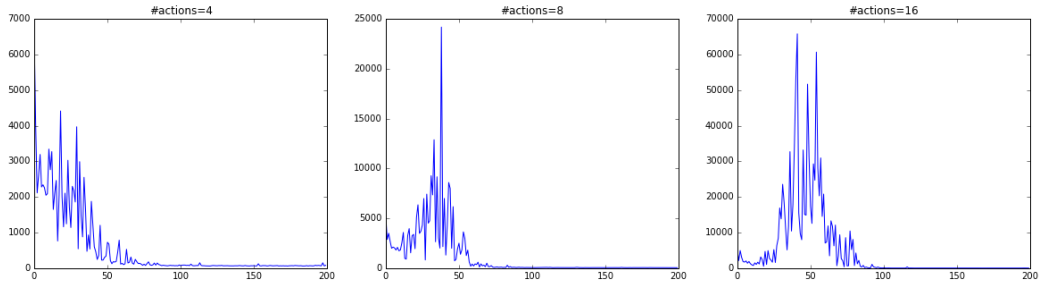

Figure 8: Learning curves for different numbers of actions. 200 trials, averaged over 5 runs, $\lambda = 0.6$.

Lastly, we also investigated the effect that the number of available choices has on the learning behavior of the agent. For that reason we set the number of available actions to m = 4, 8, 16 and simulated 5 runs with 200 trials for each case. For these simulations, we set lambda = 0.6 and again let epsilon decay exponentially over trials as illustrated in figure 8. The run-averaged escape latencies for these simulations are depicted in 8. In all cases the agent eventually optimized its action choices and learned to complete the task with a minimum amount of steps. However, the optimization converged faster with a lower number of choices. Also note the different scales of the y-axis for the three figures. In case of 4 action choices, it never took the agent more than 7000 steps to complete the task while for 8 and 16 choices this bound is at about 25000 and 68000 respectively. It is furthermore interesting to see that in the initial 10 to 20 trials, the escape latencies are much lower for the cases of 8 and 16 choices than in the subsequent approximately 50 trials. All of these observation are not unexpected and are a consequence of the particular environment and task of the experiment. In the T-maze environment are really only 4 actions required to solve the task in an optimal fashion as there are no diagonal or tilted paths. The 4 optimal motion choices are readily available in the case in which there are only 4 choices. Further action choices that are added for the 8 and 16 choices cases are therefore non-optimal and consequently the ratio of optimal-to-non-optimal choices is lowered. For example, when heading from the pick-up area to the target area, the single best option is make a step to the left. Since the agent makes random choices at the beginning, the probability of a non-optimal step is

higher for the 8 and 16 choices cases than for the case in which only 4 choices are available.

5 Acknowledgements

We would like to thank Steffen Puhlmann for helping us to debug our code. And finally, many thanks to David Higgins for supervising this project.

References

[1] Sutton, R. and Barto, A. (1998). Reinforcement learning. MIT Press, Cambridge.

[2] Watkins, C. J. C. H. (1989). Learning from Delayed Rewards. Ph.D. thesis, Cambridge University.

[3] O'Keefe, J. (1976). Place units in the hippoca mpus of the freely moving rat. Experimental neurology 51, 78-109.

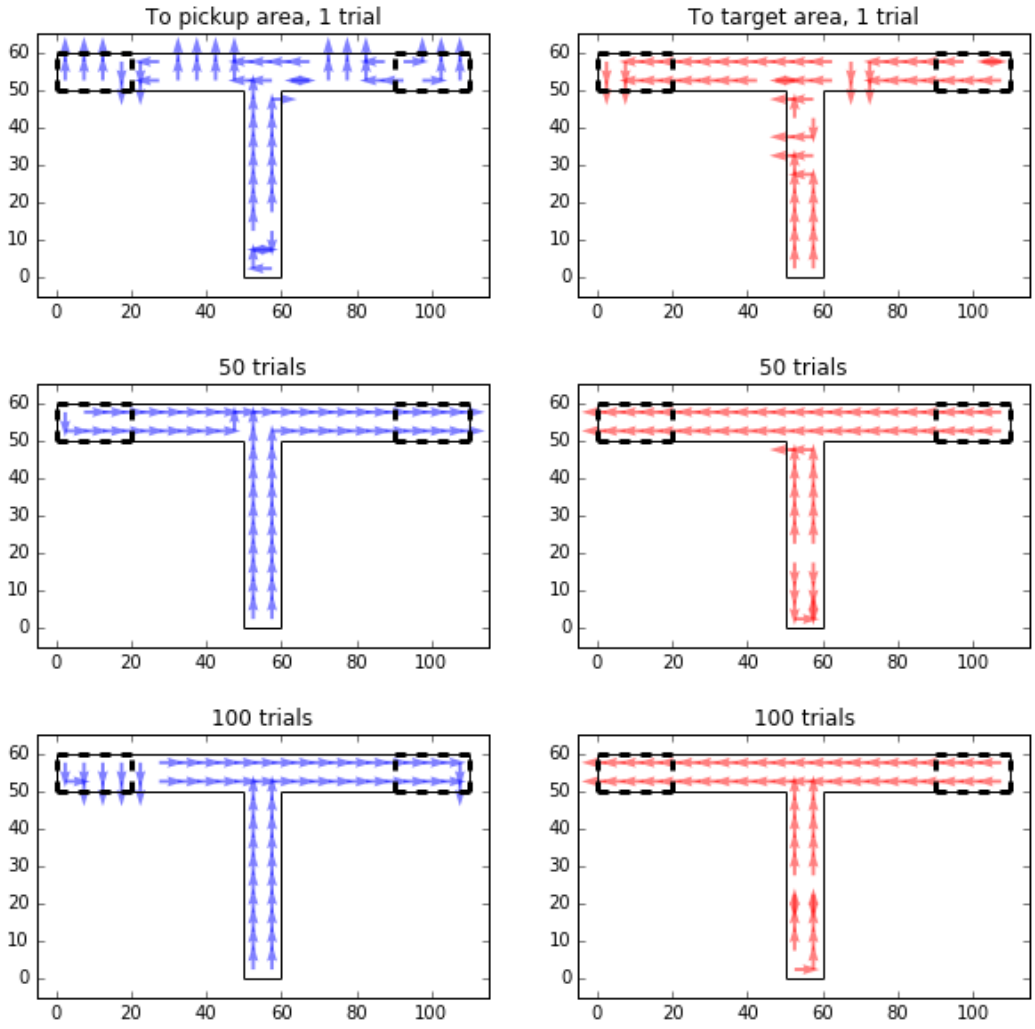

Figure 7: Navigation maps after 1, 5 and 100 trials, respectively (rows) and on the way to pickup and to target, respectively (columns).