ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Unsupervised and Reinforcement Learning in Neural Networks

Reinforcement Learning

Lecturer: Marc-Oliver Gewaltig

Teacher Assistant: Dane Corneil, Berat Denizdurduran

Students: Chiara Gastaldi and Gonçalo Cardoso Rodrigues Bonifácio Vítor

- 09-01-2015 -

Introduction

The goals of the second mini-project are two-fold: first, to implement a reinforcement learning paradigm using a rate-based neuron model where an agent gradually learns the position of a goal area by assignment of a reward and second, to study the performance of the implemented algorithm as a function of some network parameters.

Implementation of the neural network model

Decription of the setup

An agent (rat) is placed inside a quadrangular arena of unit area in a fixed position at the start of every trial ($x = 0.1, y = 0.1$). The agent should be able to learn the position of the goal area, also a fixed position in space. The state space is given by the (x, y) coordinates of the agent in the arena and it is continuous. It is used the SARSA learning algorithm with an *epsilon*-greedy policy.

When the agent collides with a wall gets a negative reward of -2 (punishment) and when it reaches the goal area, a positive reward of $+10$. The goal area is specified at the upper right corner of the arena as a 0.1-radius circle centered in ($x = 0.8, y = 0.8$) and it is hidden to the mouse.

At each position in space the agent has 8 choices of actions which correspond to moving 0.03 in one of the cardinal or intercardinal directions.

The state of the agent is encoded in the activity of a population of place cells $r_j(s)$ arranged on a grid of 20x20.

Additional parameters setting: learning rate $\eta = 0.005$, discount factor $\gamma = 0.95$, decay factor for the eligibility trace $\lambda = 0.95$, and $\epsilon = 0.5$ for the ϵ-greedy policy.

Description of the implementation

We implemented the enviroment and the neural network that controls the movement of the rat starting from the file *gridworld.py* that was provided for the exercise set 9.

Several modifications were needed to have the right specifications required in the instruction sheet:

- implementation of the gaussian activity of place cells, i.e. $r_j = \exp\left(-\frac{(x_j - s_x)^2 + (y_j - s_y)^2}{2\sigma^2}\right)$, with $\sigma = 0.05$, where s_x and s_y are the space coordinates of the mouse in the arena and x_j and y_j are the coordinates of the center of the j-th place cell;
- allowing diagonal movements;
- adapting the navigation map and updating the rule for calculating the Q-values $Q(s, a)$ to bigger number of possible directions and to the continuous state space;
- changing the point-like goal to the goal circle area.
- implementing the function integrated_reward_curve() to plot the integrated reward for each trial.
- setting a maximum number of step $Nmax$ the agent can do in on trial. If the rat doesn't reach the target in $Nmax$ steps, then the trial is aborted.

As an implementation choice, we scaled up all distances by a factor of 20, though it doesn't affect the integrity of the execution of the algorithm.

Concerning the code implementation, the vectorization of the code was a critical step in fastening the simulation process which can now take less than 10 seconds for performing 50 trials.

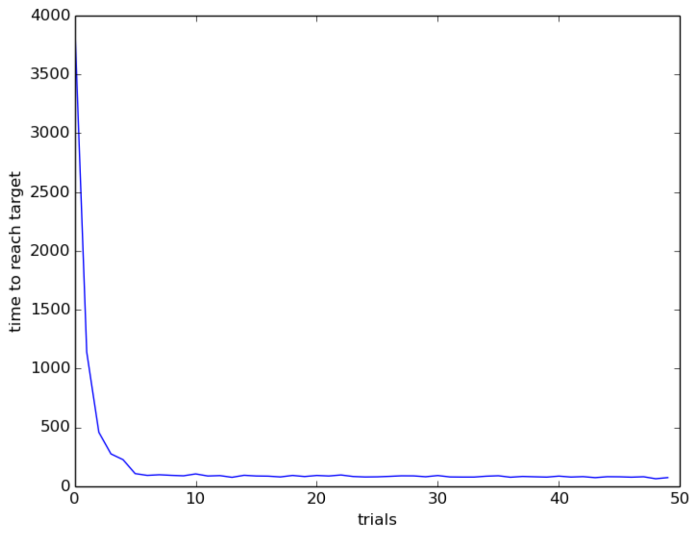

Figure 1: Learning curve averaged over 10 runs.

Analysis of the neural network model

Learning curve

We simulated 10 independent runs, each with 50 trials and acquired the respective learning curve. We chose $Nmax = 10000$ as the maximum number of steps the agent is allowed to take in one trial (to avoid infinitely long trials). Figure 1 shows the plot of the learning curve, i. e. the averaged number of time steps required for the agent to get to the goal position ("latency"). The graph has been obatained keeping $\epsilon = 0.5$ constant for all trials. The values obtained for each trial are averaged over the runs.

As expected, the average number of steps to reach the goal area decreases as the number of trials increases, reaching a plateau i.e. the agent learns the position of the reward. The plateau corresponds to an averaged steady state number of steps to reach the reward.

Integrated reward

Another way of visualizing the learning process is by considering the average total reward that was received by the agent on each trial. The plot is obtained with the function integrated_reward_curve(). The rat receives a reward of $+10$ when it enters the goal area and a reward of -2 (punishment) when it collides with one of the walls of the arena. The maximum total reward achievable is $+10$, for an agent that reaches the goal area without colliding with any wall in any of the steps of the trial (episode). As expected, and observing Figure 2, the averaged total reward increases when the number of trials increases, meaning that the agent learns to avoid the walls and moves towards the goal.

The integrated reward curve is consistent with the latency curve, while representing two complementary aspects of the learning process: in the latency curve, we observe that as several trials are taken, the agent need progressively less time (steps) to reach the goal; in the integrated reward curve, we notice the total reward increases, as the agent learns to avoid the walls (and to find the goal reward too). It should also be noted that in both curves there are plateaus corresponding to a steady-state behavior, that takes into account the ϵ-greediness of the algorithm: the agent always keeps a constant probability for wandering.

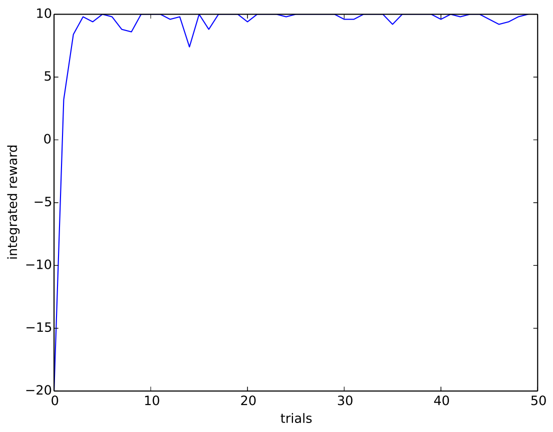

Figure 2: Total reward at each trial, averaged over 10 runs.

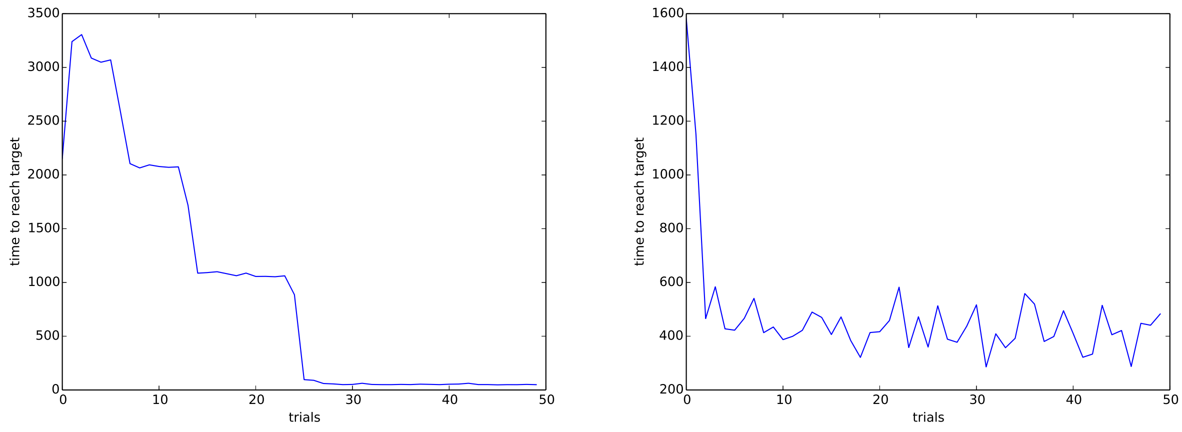

Figure 3: Learning curves for two limit cases: $\epsilon = 0.1$ (left) and $\epsilon = 0.9$ (right).

Exploration-exploitation

The parameter ϵ can vary in the range $[0, 1]$ and it represents the probability that the rat chooses a different direction than the one corresponding to the highest Q-value (ϵ-greedy policy). In other words, it is a mesure of the willingness of the rat to explore the arena. A higher ϵ corresponds to a higher probability for wandering and thus exploring new (and possibly shorter) paths to the goal position, and a lower ϵ corresponds to the exploitation of the best already found solutions, while having a low wandering probability.

In order to study how the performance of the learning process is affected, we consider $\epsilon = 0.1$ and $\epsilon = 0.9$, running again 10 runs with 50 trials.

By observing Figure 3, a few points should be noted. For $\epsilon = 0.1$, the learning curve has several plateaus corresponding to the exploitation of the best solution found so far. It takes more trials for the agent to find a competitive solution (compared to $\epsilon = 0.5$ and $\epsilon = 0.9$), but the agent can then find a solution that outperforms the other setups. For $\epsilon = 0.9$, we observe that the agent is able to find a medium solution very fast (in the first few trials), but doesn't exploit that solution, which can be seen by the high variance in the average number of required steps after reaching the plateau.

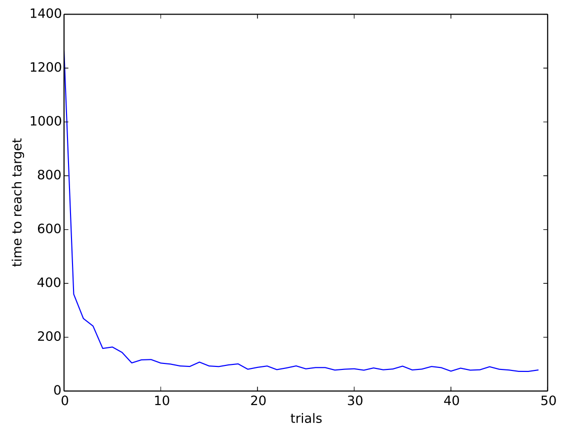

Figure 4: Learning curve for varying ϵ. 10 runs with 50 trials.

We also tried a configuration with a varying ϵ, using the following rule: $\epsilon = 0.8*(1.-trial/N\_trials)$. The results obtained are plotted in Figure 4. This strategy aims at combining both exploration (favored in the beginning, high ϵ) and exploitation (towards the end, characterized by a low ϵ).

Navigation maps

The navigation maps show the action that would be taken by the agent at each state, if following a greedy policy. For comparing the evolution of the navigation maps along the learning process, we did 1 run with 100 trials and plotted the navigation maps at the end of trials 1, 10 and 100 (Figure 5). We note that after the first trial there are still some places (states) for which the optimal action (highest Q-value) is not pointing towards the goal area (upper left corner). Between the navigation maps of trials 10 and 100 (using the default setup with $\epsilon = 0.5$) we see there is a small improvement on the latter concerning the proportion of arrows correctly pointed. Due to the eligibility trace, the propagation of information is very fast, leading to a generally good knowledge of the environment (concerning the target) in the very first trials. We should also note that even for 100 trials the regions on the upper left and lower right corners are poorly organized as the rat already has a very large corridor to go to the goal area. To achieve better navigation maps, the starting position of the rat should be random, so that all the regions of the arena would be more evenly explored and learnt.

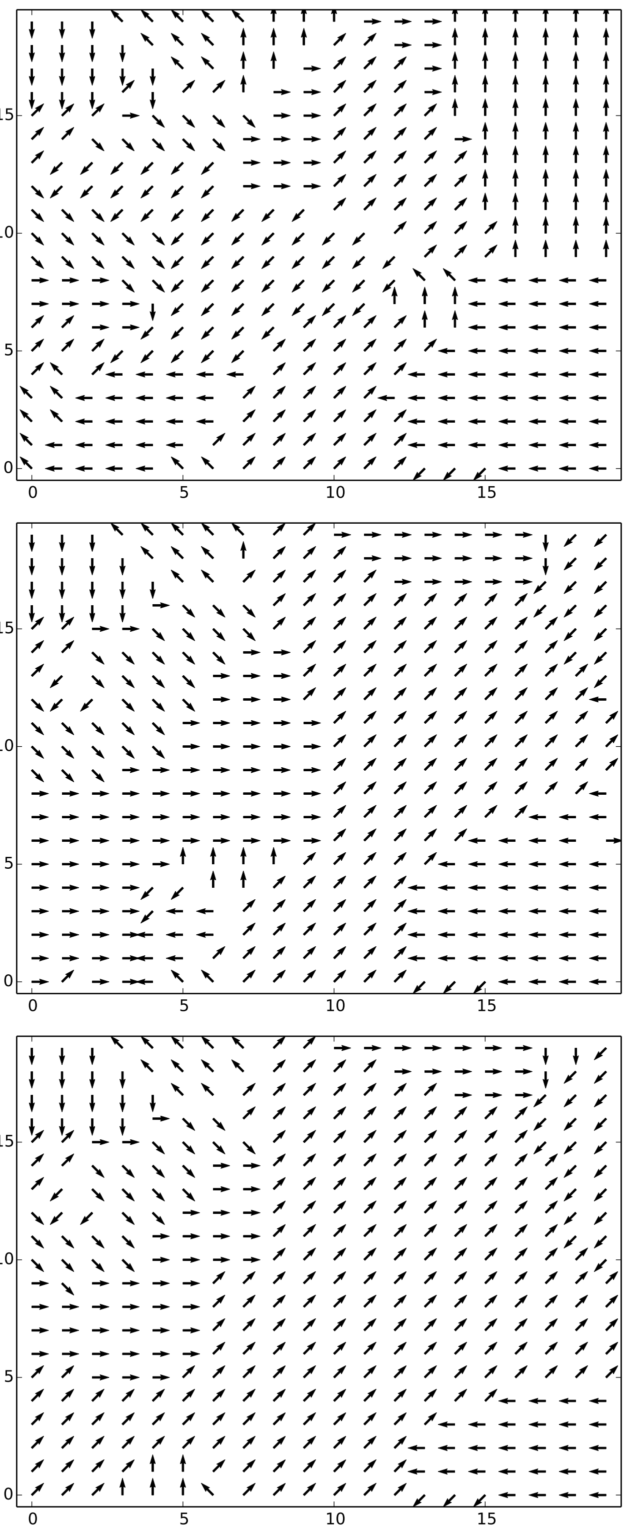

Figure 5: Navigation maps for trials 1 (up), 10 (center) and 100 (lower) of the same run.