University: Sharif University of Technology

Department: Electrical Engineering

Course Name: Advanced Neuroscience

Homework 6 Report

Reinforcement Learning

Student Name: Ali Shahbazi

Student ID: 98101866

Instructor: Prof. Ali Ghazizadeh

Academic Semester: 2023 Spring

Implementation

- **Simple RL Model Free**

  In this method I used values of cells as the main parameter for deciding next step and learning. Note that r_t is omitted from update equation because the value of targets are set initially.

  - Update Rule:

$$V(S_t)_{new} = V(S_t)_{old} + \eta\delta_t$$
$$\delta_t = \gamma V(S_{t+1}) - V(S_t)$$

  - Transition Rule:

$$P(a_i|S_t) = \frac{\exp(\beta m_i)}{\sum_j \exp(\beta m_j)}$$
$$m_i = V(S_{t+1}|a_i)$$

  Above equations indicate that neighbors with higher value, are more likely to be chosen. Also, by transition of state we update the value of last cell by using a simple difference rule.

  The value of β is critical in order to let the mouse move freely and explore the map, or act greedy and use only its previous knowledge. To implement more realistic experiment, we change the value of β in each trial by a sigmoid function described in Figure 1. Lower value of β in first trials, let the mouse explore the map and find different targets, while higher values of β forces the mouse to act extremely greedy and find home in shorter steps.

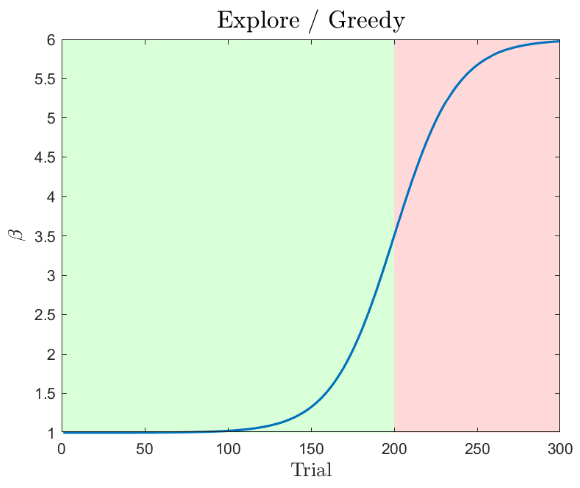


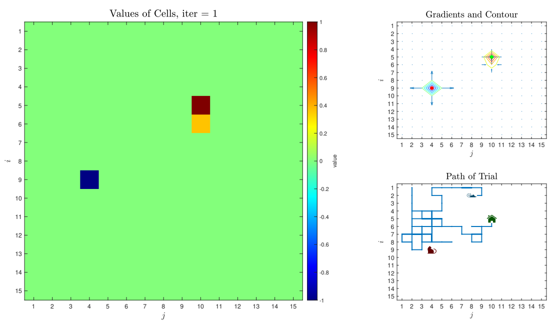
Figure 1: Value of β over Trials

$$\beta(x) = 1 + \frac{5}{1 + \exp(-(x - \frac{2T}{3})/\frac{T}{16})}$$

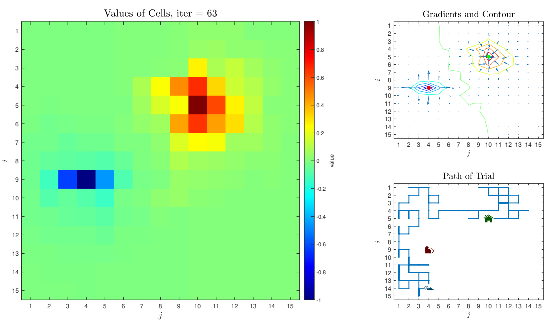
- **TD(λ)**
  Here we use these new formulas:
  - Update Rule:
  $$V(S_t)_{new} = V(S_t)_{old} + \eta\delta_t M$$
  $$\delta_t = \gamma V(S_{t+1}) - V(S_t)$$
  And M, is a matrix that all elements are zero, except those elements representing mouse path with a factor λ. For example if our map is 3×3 and we started from $(1,1)$, then $(1,2)$ and ended in $(2,2)$, then $(3,2)$ then M would be:
  $$M = \begin{bmatrix} \lambda^3 & \lambda^2 & 0 \\ 0 & \lambda & 0 \\ 0 & 1 & 0 \end{bmatrix}$$
  - Transition Rule (same as previous):
  $$P(a_i|S_t) = \frac{\exp(\beta m_i)}{\sum_j \exp(\beta m_j)}$$
  $$m_i = V(S_{t+1}|a_i)$$

Questions

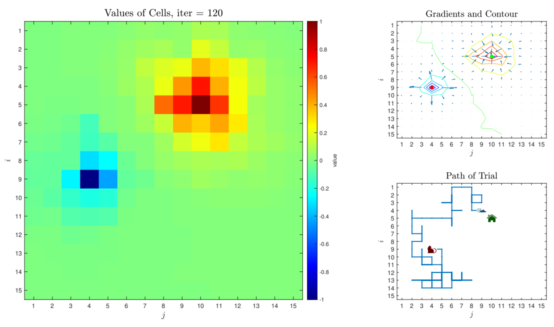
1) Figure 2 shows snapshots of values, contours, gradients and path of trials. Clearly it can be seen that value of target is propagating and arrows show which direction mouse is likely to choose.



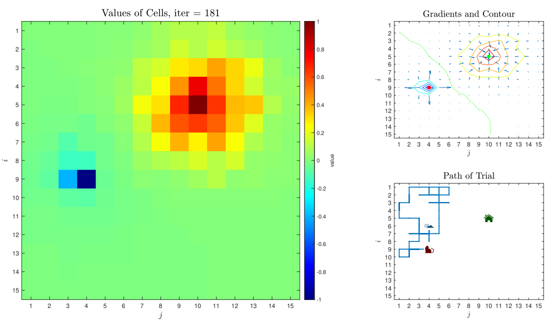
(a) Trial 1



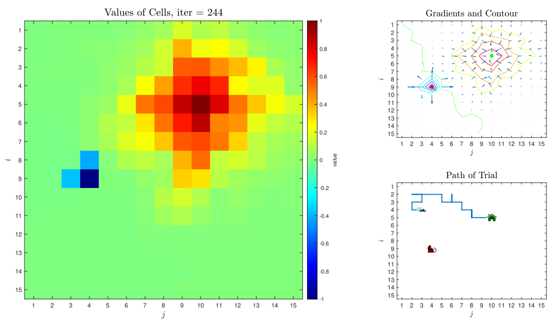
(b) Trial 63



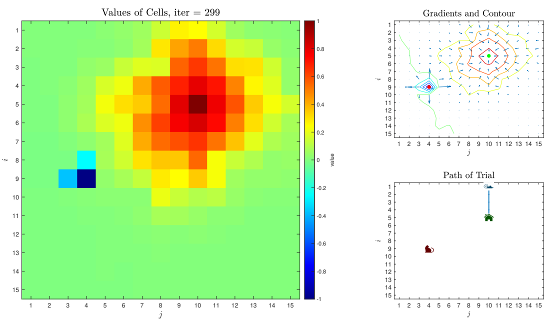
(c) Trial 120



(d) Trial 181



(e) Trial 244



(f) Trial 299

Figure 2: Plots of Different Trials During Learning - TD - One-Target

*Note: File named `demo1.avi` is attached.

2) Contours and gradients are plotted in Figure 2. In addition, we can run this simulation 100 times and calculate average steps needed to reach the target (home vs. cat). According to Figure 3, after 150 trials we can see a noticeable decay in needed steps and after 200 trials by entering greedy mode, needed steps has decreased and then converged to a value around 45. Note that this simulation is run for $\eta = 0.4$ and $\gamma = 0.9$.

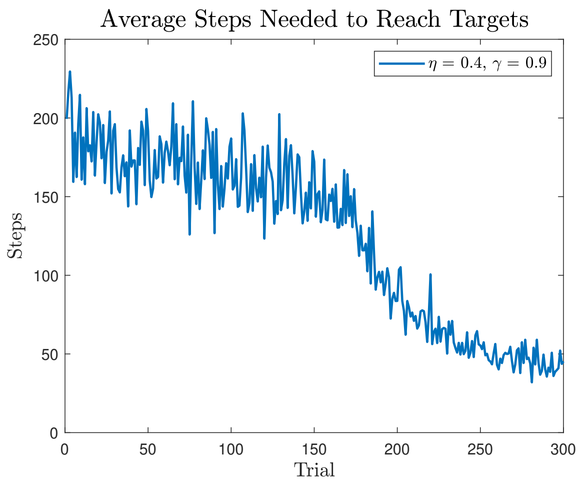


Figure 3: Average Steps

3) In this part, we change the value of η and γ over [0.20.40.60.8] values. In Figure 4, c-axis in log-scale shows average steps needed to reach the target in the last 50 trials. As we can see, best values for parameters is likely to be around $\eta = 0.4$ and $\gamma = 0.8$. This is done for 10 repetitions and mean is reported.

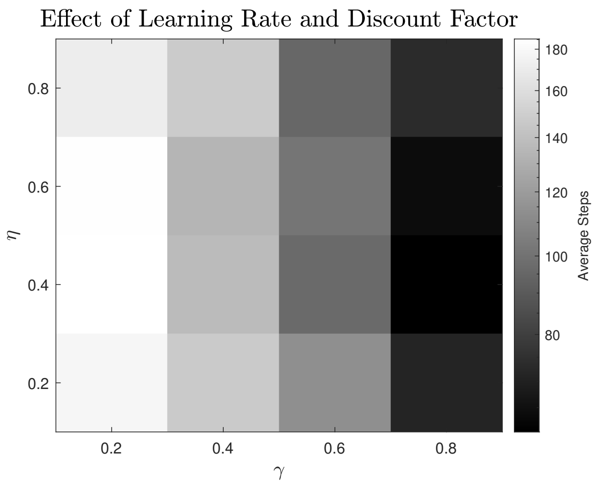
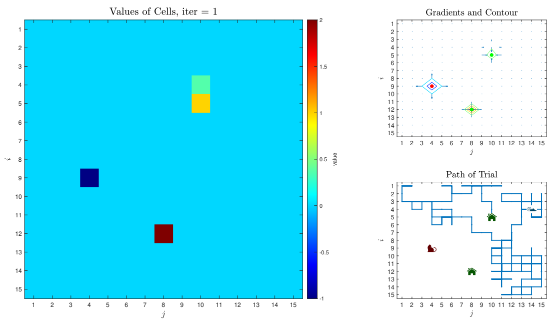


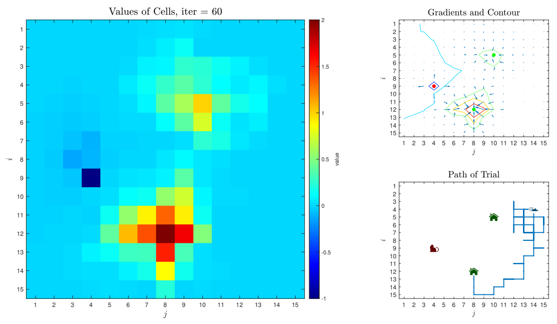
Figure 4: Effect of η and γ

By setting *discount factor* (γ), we choose how much proportion of next state value we want to take, and by setting *learning rate* (η), we choose the strength of learning and updating values in each step. In real world, learning rate can be a function of time, probably decreasing over time because the uncertainty decreases.

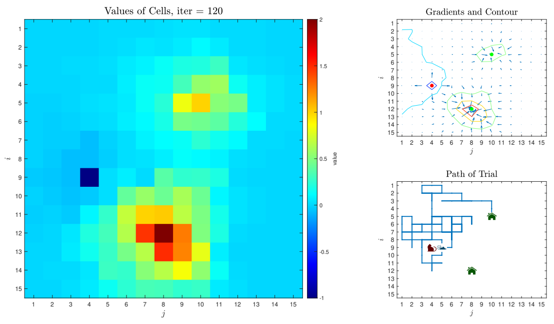
4) Here we add another target with higher value (2) to the map. Figure 5 shows snapshots of values, contours, gradients and path of trials. Learning is noticeable.



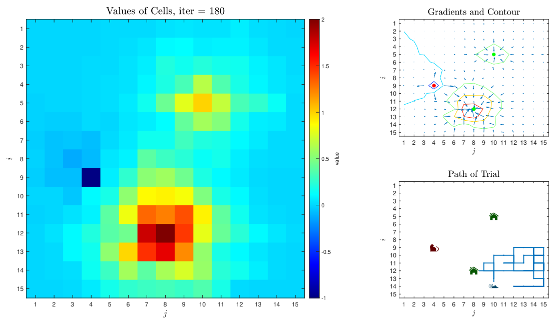
(a) Trial 1



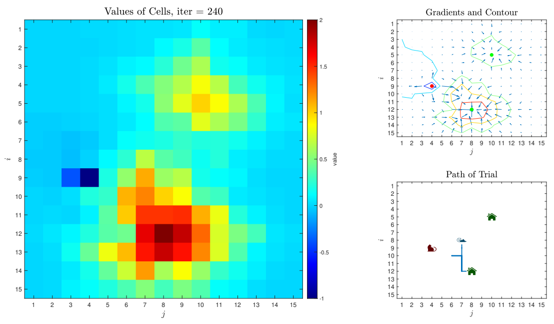
(b) Trial 60



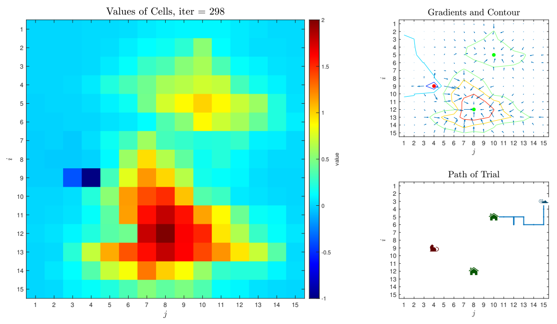
(c) Trial 120



(d) Trial 180



(e) Trial 240



(f) Trial 298

Figure 5: Plots of Different Trials During Learning - TD - Two-Target

*Note: File named `demo2.avi` is attached.

And we change the value of η and γ over [0.20.40.60.8] values again for two targets. In Figure 6, c-axis in log-scale shows average steps needed to reach the target in the last 50 trials. As we can see, best values for parameters is likely to be around $\eta = 0.6$ and $\gamma = 0.8$. This is done for 10 repetitions and mean is reported.

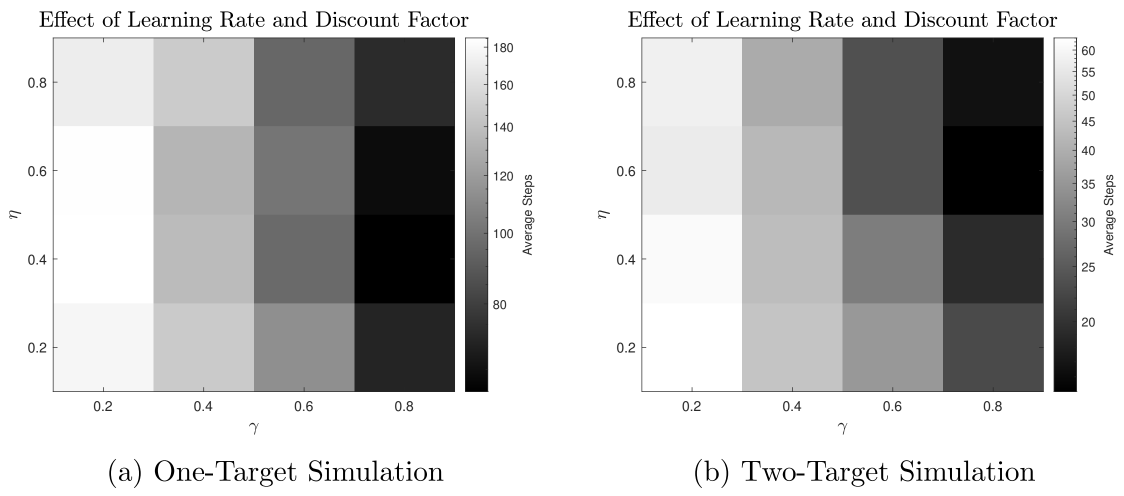
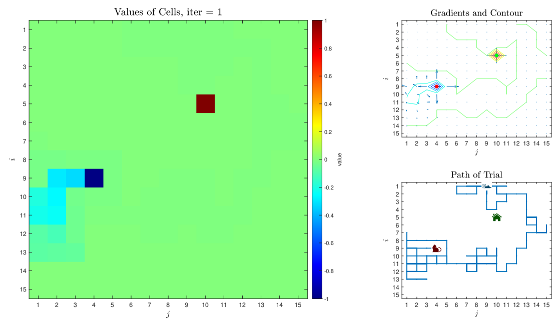


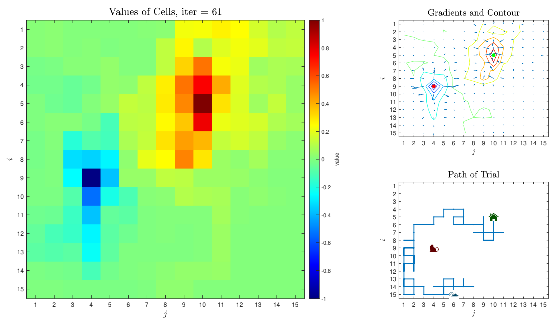
Figure 6: Comparing Effect of η and γ

It seems that higher learning rates now can be better, but the most important point in this figure is the scale of c-axis. Clearly, the average steps needed to reach the target has decreased severely, due to adding second target and same map size.

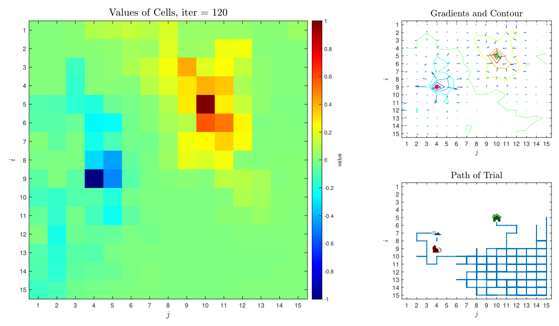
5) Now using method explained in *Implementation*, we train the mouse with $TD(\lambda)$ rule. Figure 7 shows snapshots of values, contours, gradients and path of trials. As can be seen, $TD(\lambda)$ is much faster than simple RL method. Note that here $\lambda = 0.9$.



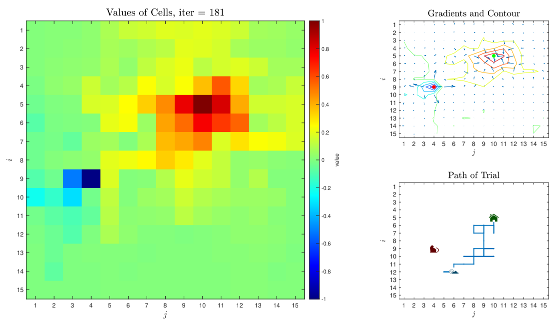
(a) Trial 1



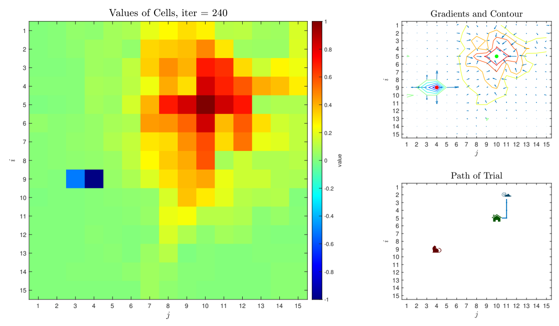
(b) Trial 61



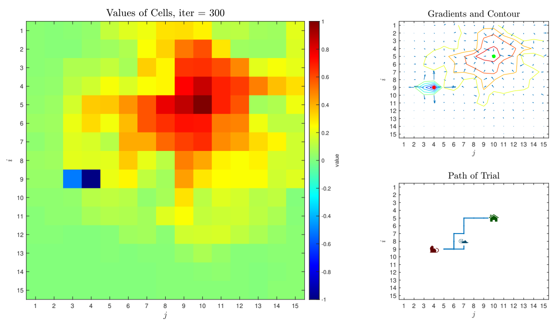
(c) Trial 120



(d) Trial 181



(e) Trial 240



(f) Trial 300

Figure 7: Plots of Different Trials During Learning - $TD(\lambda)$ - One-Target

*Note: File named `demoLambda1.avi` is attached.

If we run the simulation 100 times and calculate average steps needed to reach the target (home vs. cat), Figure 8 would appear. Surprisingly, there is not a significant difference between performance of these methods, but we should keep in mind that we change the value of β over time and decide where to start acting greedy. So, if we set the β as a constant, then we can compare performances correctly.

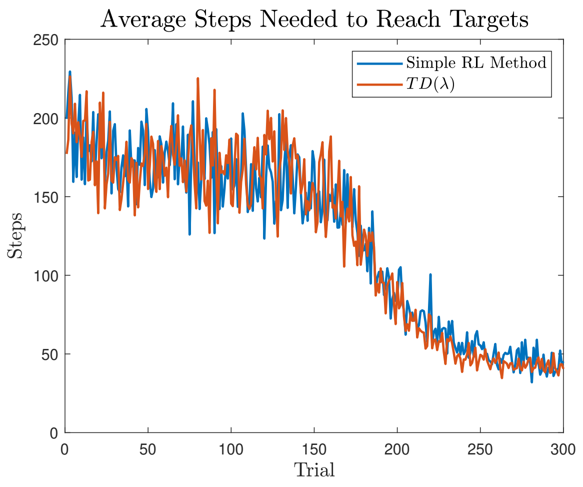


Figure 8: Average Steps

And we can change the value of λ to see the difference. Figure 9 shows best values for λ.

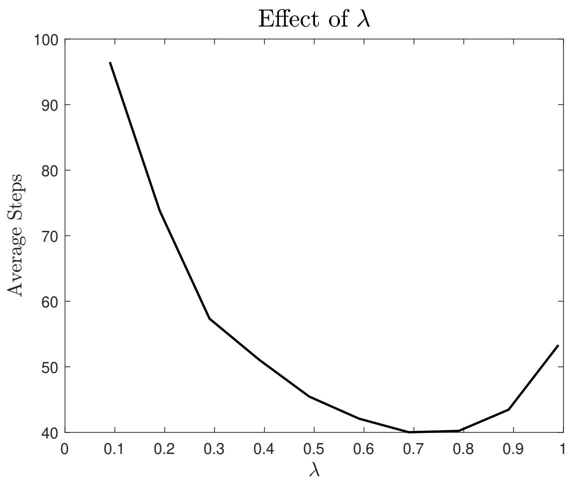
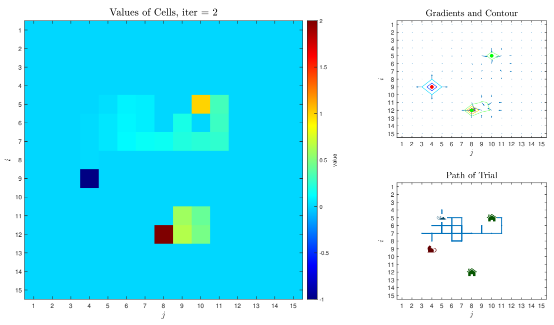
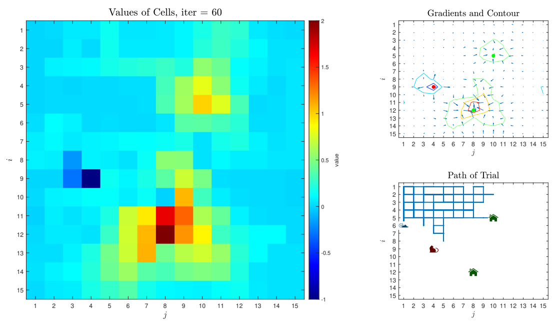


Figure 9: Effect of λ on Training Speed

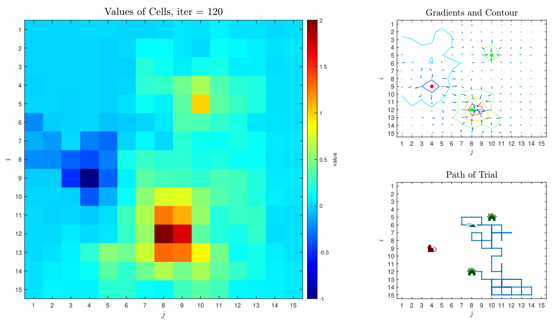
Here we do same steps for two-target situation.



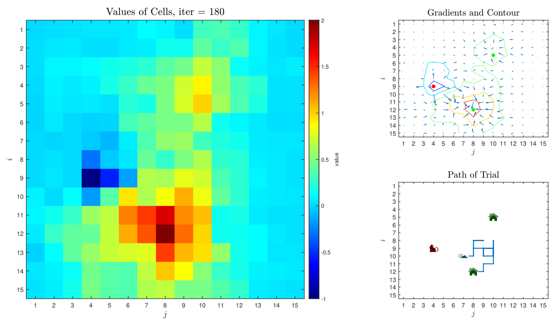
(a) Trial 2



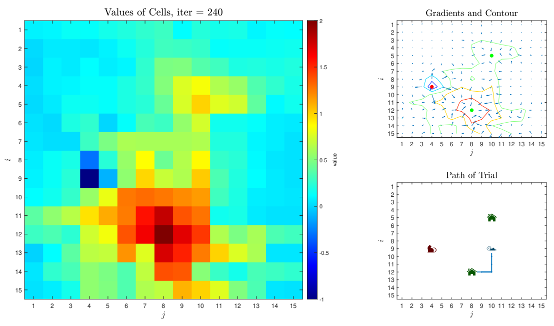
(b) Trial 60



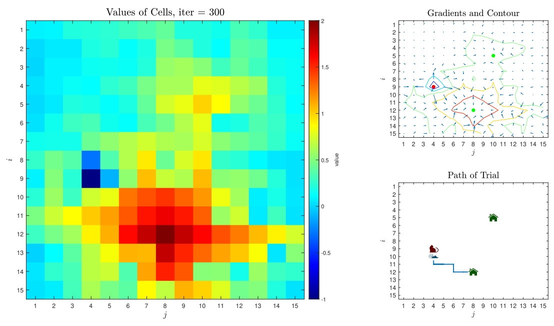
(c) Trial 120



(d) Trial 180



(e) Trial 240



(f) Trial 300

Figure 10: Plots of Different Trials During Learning - $TD(\lambda)$ - Two-Target

*Note: File named `demoLambda2.avi` is attached.

And we can change the value of λ to see the difference. Figure 9 shows best values for λ.

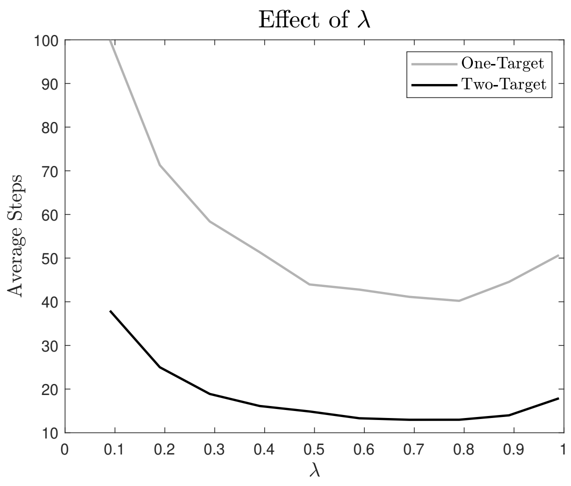


Figure 11: Comparing Effect of λ on Training Speed