Reinforcement Learning HW 6

Mert Bilgin (7034879)
mert.bilgin@student.uni-tuebingen.de

Lalitha Sivakumar (6300674)
lalitha.sivakumar@student.uni-tuebingen.de

Kevin Van Le (7314700)
kevin-van.le@student.uni-tuebingen.de

November 27, 2025

1 Function approximation

(a)

Tabular methods are a special case of linear function approximation. For a finite state space with n states, each state s is represented by a one-hot feature vector $\phi(s) \in \mathbb{R}^n$ defined by

$$\phi(s)_i = \begin{cases} 1, & \text{if } i \text{ corresponds to state } s, \\ 0, & \text{otherwise.} \end{cases}$$

With linear value function approximation,

$$\hat{V}(s) = w^\top \phi(s),$$

the one-hot structure ensures that $\hat{V}(s)$ simply selects the parameter w_s. Thus tabular methods are exactly linear function approximators with one-hot features.

(b)

For a continuous state $s = (s_1, \ldots, s_k) \in \mathbb{R}^k$, an order-$n$ polynomial feature has the form

$$x_i(s) = \prod_{j=1}^{k} s_j^{c_{i,j}},$$

where each exponent $c_{i,j}$ is an integer in $\{0, 1, \ldots, n\}$.
Each dimension j therefore has $(n+1)$ possible exponent choices, and the choices across the k dimensions are independent. Hence the total number of distinct polynomial features is

$$(n+1)^k.$$

2 Feature Designing

A simple feature vector for a linear value function in Super Mario could be

$$x(s) = (x_1, x_2, x_3, x_4, x_5),$$

where each entry just picks out something important from the current frame:

- x_1: Mario's current x-position in the level.
- x_2: his height (ground, platform, or in the air).
- x_3: distance to the closest enemy in front of him.
- x_4: whether the next tile is safe to step on (1) or not (0).
- x_5: his power-up state (small / super / fire).

3 Value Function Fitting

(a)

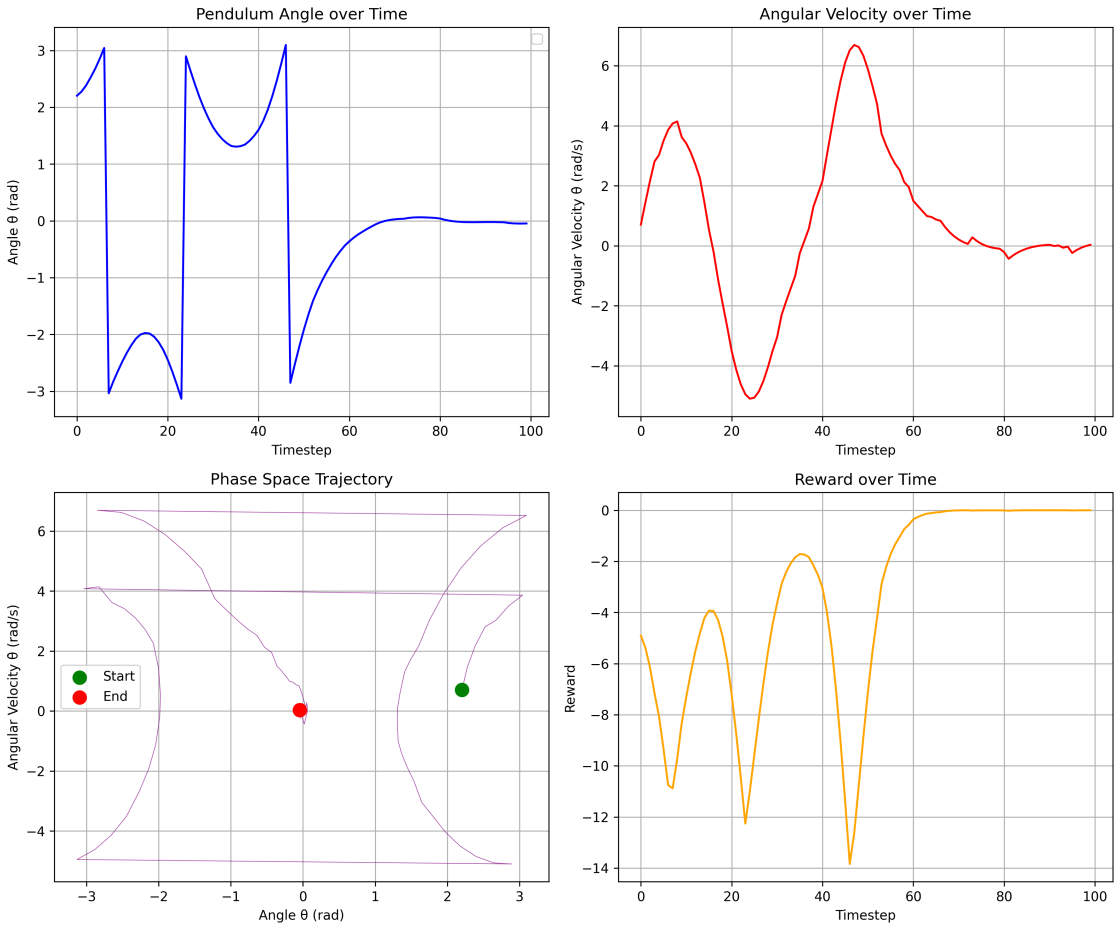

Figure 1: Trajectory of the pendulum system.

(b)

Figure 2 shows the learning progress over 2000 iterations with $\gamma = 0.95$. The loss decreases initially but becomes noisier and starts to converge around the 1300th iteration.

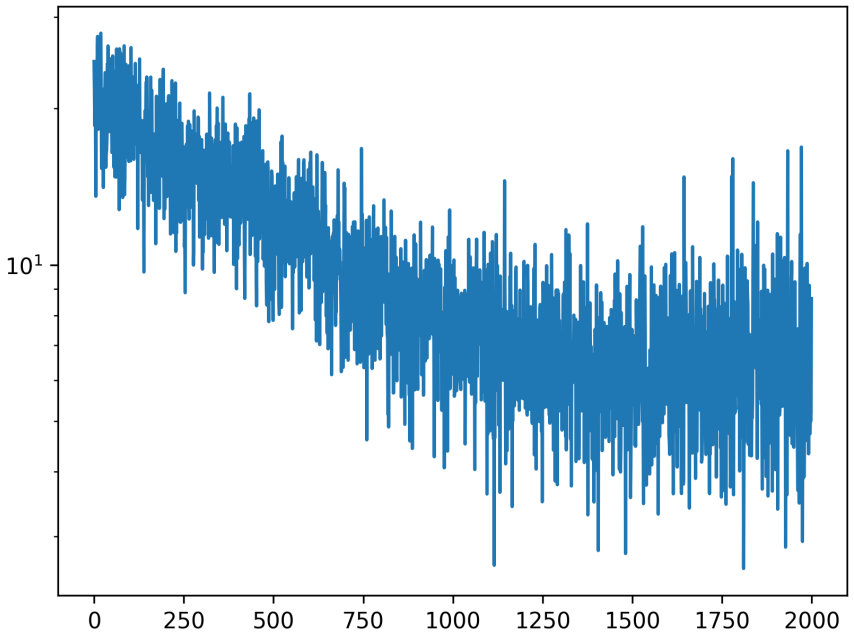

Figure 2: Learning curve for $\gamma = 0.95$ (log scale).

(c)

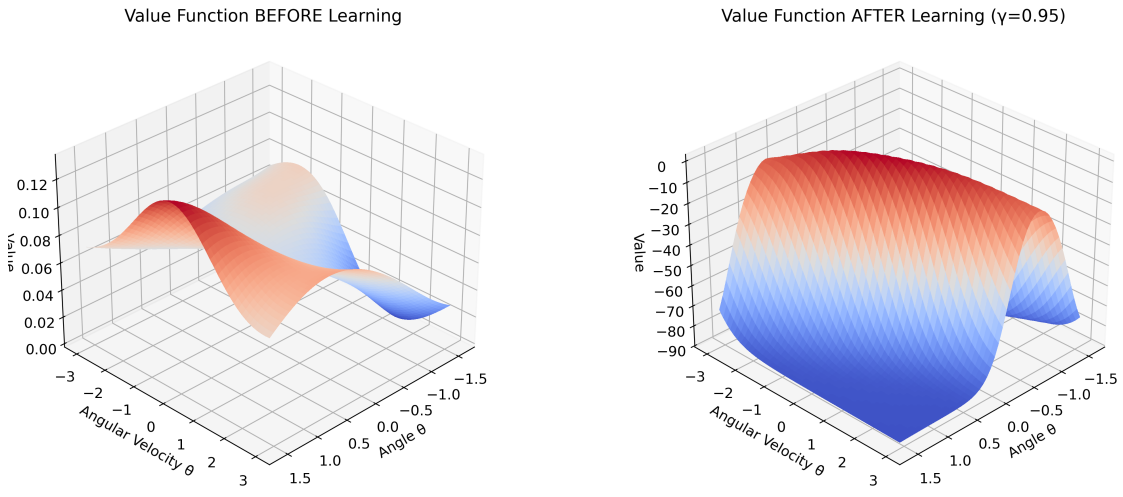

Figure 3: Value function before and after learning.

(d)

The value function has its maximum near the upright position ($\theta \approx 0$) with low angular velocity. This makes sense because states close to the upright position with low velocity are easy to keep stable and get higher returns. States far from upright or with high velocity need more control effort and get more negative rewards, so they have lower values. The value function is symmetric around $\theta = 0$ because of the symmetric pendulum dynamics.

(e)

Figure 4 compares the learning curves for $\gamma = 0.95$ and $\gamma = 0.5$. The $\gamma = 0.5$ case converges to much lower loss values, likely because the TD targets have lower variance due to immediate rewards being weighted a lot more, so therefore the value function is easier to approximate. The value functions look very different (Figure 5). With $\gamma = 0.5$, the agent cares much more about near-term rewards, so the values don't diverge due to less accumulation of good / bad future rewards.

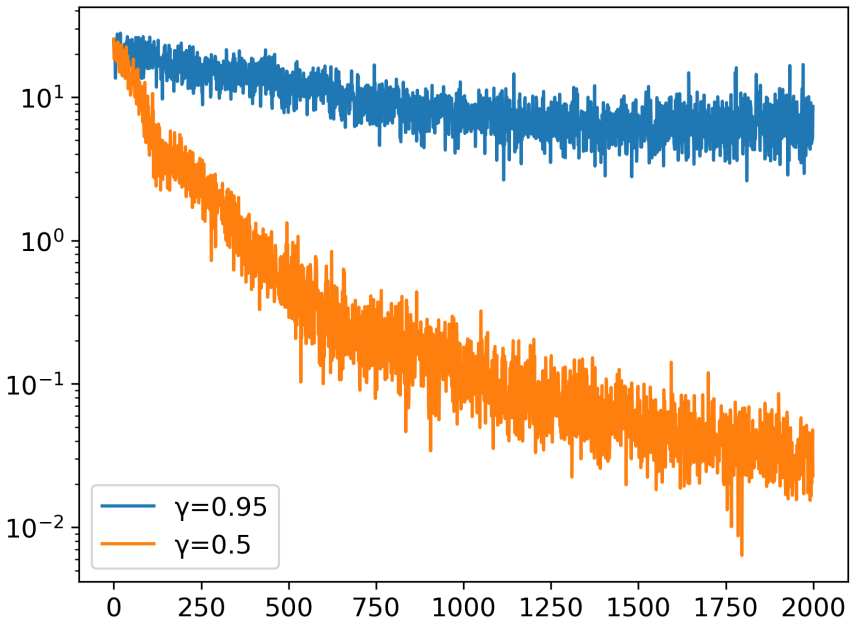

Figure 4: Learning curves for $\gamma = 0.95$ and $\gamma = 0.5$.

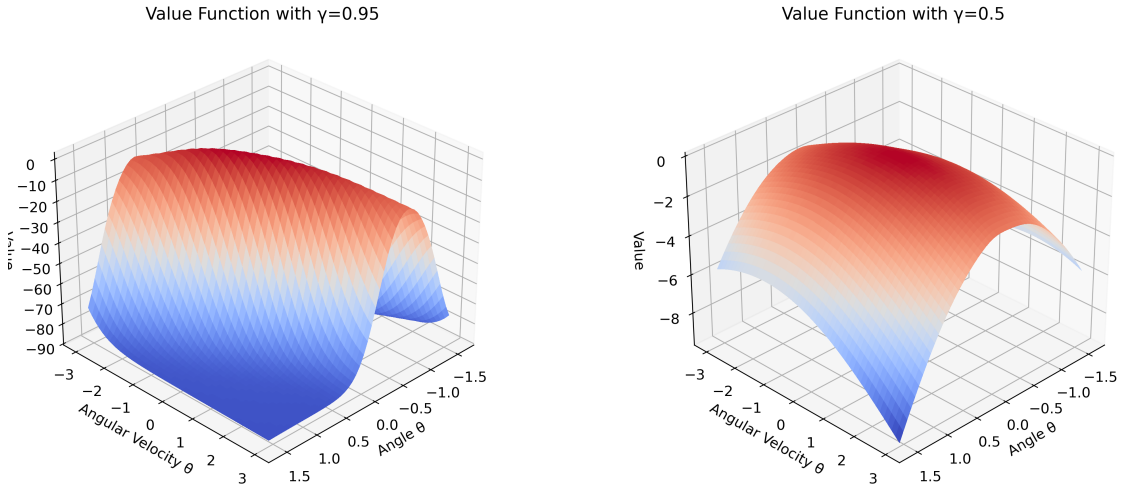

Figure 5: Value functions with $\gamma = 0.95$ (left) and $\gamma = 0.5$ (right).