Reinforcement Learning: Homework 3

Jorrim Prins
Student nr.: 11038934
jorrimprins.prins@student.uva.nl

Yke Rusticus
Student nr.: 11306386
yke.rusticus@student.uva.nl

5.3 Gradient Descent Methods

1. An unbiased estimate of the value in the current state can be obtained by looking at the Monte Carlo data. As MC data is generated by interaction with the environment using policy π, importance weights are not necessary to begin with. Now the true value of a state is the expected value of the return following from the state and the MC target G_t is by definition an unbiased estimate of $v_\pi(S_t)$

2.

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2}\alpha\nabla\left[R_{t+1} + \hat{v}\left(S_{t+1}, \mathbf{w}_t\right) - \hat{v}\left(S_t, \mathbf{w}_t\right)\right]^2 \\ &= \mathbf{w}_t - \alpha\left[R_{t+1} + \hat{v}\left(S_{t+1}, \mathbf{w}_t\right) - \hat{v}\left(S_t, \mathbf{w}_t\right)\right] \\ &\quad \cdot \left(\gamma\nabla\hat{v}\left(S_{t+1}, \mathbf{w}_t\right) - \nabla\hat{v}\left(S_t, \mathbf{w}_t\right)\right)\end{aligned} \tag{1}$$

The origin of the "Semi" in Semi-Gradient TD is very obvious from the derivation above. If Gradient Descent would be applied to the Mean Squared Temporal Difference error, the gradient would exist of two separate components as the MSTD error has two components dependent on w_t ($\gamma\nabla\hat{v}\left(S_{t+1}, \mathbf{w}_t\right)$ and $\nabla\hat{v}\left(S_t, \mathbf{w}_t\right)$). Semi-Gradient TD only takes into account the latter part of the gradient (the error multiplied by $-\nabla\hat{v}\left(S_t, \mathbf{w}_t\right)$) and therefore only uses part of the full gradient, which is why it is called "Semi".

3. One advantage of bootstrapping is that it typically enables significantly faster learning. However, in the Mountain Car problem, the enabling of continual and online learning is the important aspect that gives preference to bootstrapping over MC methods. As MC only updates after completion of an episode, which might take a while, bootstrapping would probably learn way faster as it allows for continual and within-episode learning.

6.2 Geometry of linear value-function approximation

1.

$$\begin{aligned}\bar{\delta}_w(0) = B^\pi v_w - v_w &= \mathbf{E}_\pi\left[R_{t+1} + \gamma v_w(S_{t+1}) - v_w(S_t) | S_t = 0, A_t \sim \pi\right] \\ &= 0 + \gamma w\phi_1 - w\phi_0 = 0 + 1 - 2 = -1\end{aligned} \tag{2}$$

$$\begin{aligned}\bar{\delta}_w(1) = B^\pi v_w - v_w &= \mathbf{E}_\pi\left[R_{t+1} + \gamma v_w(S_{t+1}) - v_w(S_t) | S_t = 1, A_t \sim \pi\right] \\ &= 0 + \gamma w\phi_0 - w\phi_1 = 0 + 2 - 1 = 1\end{aligned} \tag{3}$$

So $\bar{\delta}_w(s) = \begin{bmatrix}-1 \\ 1\end{bmatrix}$.

2. $MSBE = 0.5 \cdot -1^2 + 0.5 \cdot 1^2 = 1^2 = 1$

3. Our target from 6.2.1 (with $w = 1$, $\gamma = 1$ and zero rewards) is:

$$B^\pi v_w = \begin{bmatrix} R_{t+1} + \gamma(v_w)_1 \\ R_{t+1} + \gamma(v_w)_0 \end{bmatrix} = \begin{bmatrix} 0 + \gamma \cdot w \cdot \phi_1 \\ 0 + \gamma \cdot w \cdot \phi_0 \end{bmatrix} \tag{4}$$

$$= \begin{bmatrix} 0 + 1 \cdot 1 \cdot 1 \\ 0 + 1 \cdot 1 \cdot 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{5}$$

So now we aim to find:

$$w^* = \arg\min_w \left\| \begin{bmatrix} 1 \\ 2 \end{bmatrix} - v_w \right\|^2 \tag{6}$$

$$= \arg\min_w \left\| \begin{bmatrix} 1 \\ 2 \end{bmatrix} - w \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} \right\|^2 \tag{7}$$

$$= \arg\min_w \left\| \begin{bmatrix} 1 - w \cdot 2 \\ 2 - w \cdot 1 \end{bmatrix} \right\|^2 \tag{8}$$

$$= \arg\min_w (1 - 2w)^2 + (2 - w)^2 \tag{9}$$

Find derivative and set to zero:

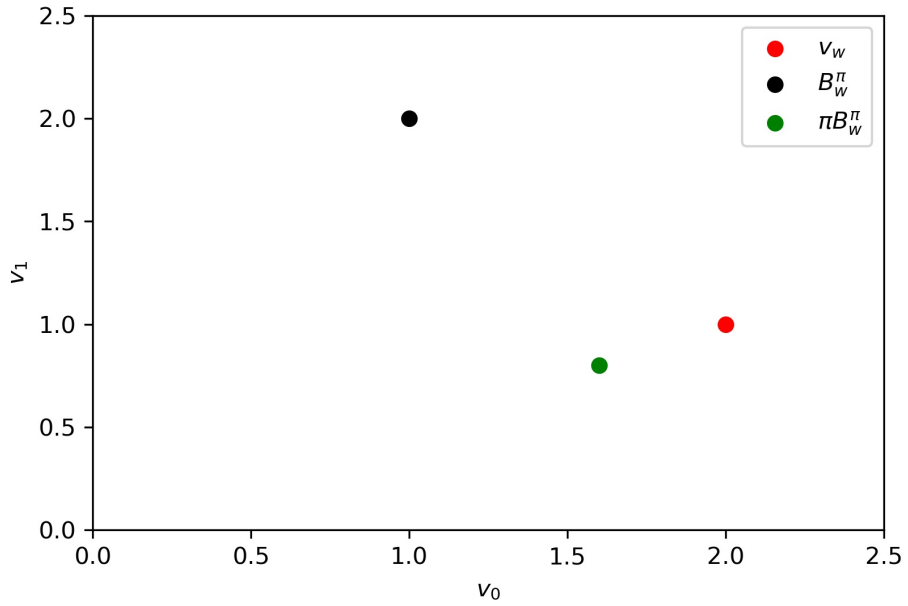
$$\frac{\partial}{\partial w} \left((1 - 2w)^2 + (2 - w)^2 \right) = -4(1 - 2w) - 2(2 - w) \tag{10}$$

$$= -4 + 8w - 4 + 2w = -8 + 10w \stackrel{!}{=} 0 \tag{11}$$

$$10w^* = 8 \tag{12}$$

$$w^* = \frac{4}{5} \tag{13}$$

4. The figure below shows the value function for initial weight $w = 1$ as the red dot. The black dot represents the projection by Bellman transformation of the value vector into the larger space, where πB^π_w (the green dot) then represents the projection of this transformation back onto the smaller subspace. This point is equal to the value function under the "optimal" weight calculated in the previous question, so closest to B^π_w in least-squares sense.



6.4 Coding Assignment - Deep Q Networks

2. We could still use a tabular approach if we divide our infinite state space into finite bins. For example, if a state s could have any value between 0 and 1, we have have an infinite amount of possible states. If we want to use a tabular approach we could divide that state space into the bins $s \leq 0.5$ and $s > 0.5$. The larger the bins, the more you (over)simplify the problem. The smaller the bins, the larger your state space, but the more accurate you can learn. In the case where we need a large amount of precision, using a tabular approach is infeasible. For example, if we would have a very large cliff-world where an obstacle is placed between the goal and the agent, we risk overlooking the obstacle if we choose our bins too large. In that case you might end up needing such small bins that a tabular approach is infeasible. In our CartPole problem we do not have such obstacles and we do not need very large precision to learn the game, so a tabular approach could be used in this case.