UNIVERSITEIT VAN AMSTERDAM
Faculteit Natuurwetenschappen, Wiskunde en Informatica

Reinforcement Learning Week 4

By

Sergio Alejandro Gutierrez Maury, 11821353
Ilse Feenstra, 13628356

October 13, 2023

Homework: Geometry of linear value-function approximation (Application)

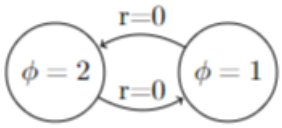

Figure 1: Figure 2 from Assignment

Consider the two-state MDP given in Figure 1. It consists of two states with one action, that transition into one another with reward 0. The features for both states are $\phi = 2$ for state s_0 and $\phi = 1$ for state s_1. We will now predict the value of the states using $v_w = w \cdot \phi$

1. We can write the Bellman error vector as

$$\bar{\delta} = B^{\pi} v_w - v_w \tag{1}$$

where B^{π} is the Bellman operator. What is the bellman error vector after initialization with $w = 1$ and using $\gamma = 1$?

The Bellman operator B^{π} for a policy π is defined as:

$$B^{\pi} v_w(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')]$$

Using $\gamma = 1$ and given that there is only one action and the reward is 0, we have:

$$B^{\pi} v_w(s) = v(s')$$

The value function is $v_w = w \cdot \phi$, with $w = 1$, we then have $v_w = \phi$. Therefore, the value states for each state would be:

$$v_w(s_0) = 2$$

$$v_w(s_1) = 1$$

Then, the errors are:

$$\delta(S_0) = B^{\pi}(s_0) - v_w(s_0) = 1 - 2 = -1$$

$$\delta(S_1) = B^{\pi}(s_1) - v_w(s_1) = 2 - 1 = 1$$

and the vector, $\bar{\delta} = (-1, 1)$

2. What is the Mean Squared Bellman Error?

The Mean Squared Bellman Error equation is:

$$\overline{BE}(\mathbf{w}) = \left\|\bar{\delta}_{\mathbf{w}}\right\|_{\mu}^{2}$$

Assuming that both states are equally visited, we have $\mu(s_0) = \mu(s_1) = \frac{1}{2}$. Then the formula becomes:

$$\overline{BE} = \frac{1}{2} \times (-1)^2 + \frac{1}{2} \times (1)^2 = 1$$

3. What w results in the value functions that is closest (in least-squares sense) to the target values $B^{\pi}v_w$? Use the target values you found in 1.

To find the w that results in the value function that is closest to the target values $B^{\pi}v_w$, we need to minimize the mean-squared projected Bellman error, so we need to find:

$$\mathbf{w} = argmin_{\mathbf{w}\in R^d} \|B^{\pi}v_w - v_w\|_{\mu}^{2}$$

Using results found in 1:

$$\mathbf{w} = argmin_{\mathbf{w}\in R^d} \left\| \begin{pmatrix} 1 \\ 2 \end{pmatrix} - \begin{pmatrix} 2 \\ 1 \end{pmatrix} \cdot \mathbf{w} \right\|_{\mu}^{2}$$

Using again $\mu(s_0) = \mu(s_1) = \frac{1}{2}$:

$$\mathbf{w} = argmin_{\mathbf{w}\in R^d} (\frac{1}{2} \cdot (1-2w)^2 + \frac{1}{2} \cdot (2-w)^2)$$

Simplifying and taking derivative:

$$= \frac{5}{2}w^2 - 4w + \frac{5}{2}$$

$$\frac{d}{dw}(\frac{5}{2}w^2 - 4w + \frac{5}{2}) = 5w - 4 = 0$$

Solving for $\mathbf{w}$, we find that the value of $\mathbf{w}$ that is closest to the target values $B^{\pi}v_w$ is $\frac{4}{5}$.

4. Plot v_w, $B^\pi v_w$, and $\prod B^\pi v_w$. Explain what is happening. *(Hint: Refer to Figure 11.3 in the book).*

Replacing the **w** we found in the previous exercise, we find that projection $\prod B^\pi v_w$ is:

$$\prod B^\pi v_w = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \frac{4}{5} = \begin{pmatrix} \frac{4}{5} \\ \frac{8}{5} \end{pmatrix}$$

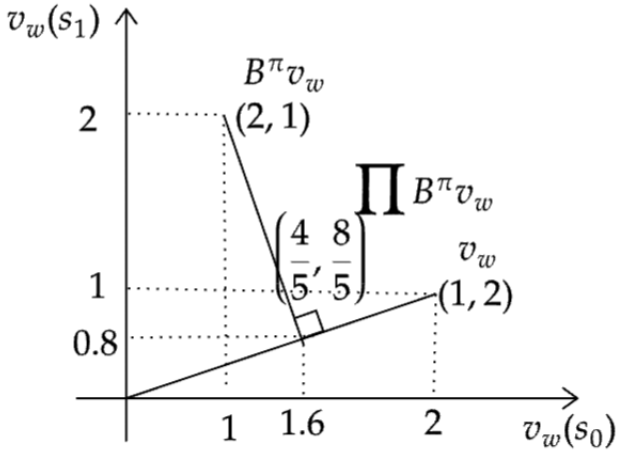

Figure 2: Plotting values

This figure depicts the value function v_w when $w = 1$ in a two-dimensional space. The line through v_w and the origin represents the subspace of all value functions. When the weight of question 3 is applied, you get B_w^π. However, this point is not in the subspace of the value function. Therefore, we use the projection of B_w^π to get B_w^π. This projection is orthogonal to the line.

Homework: Coding Assignment - Deep Q Networks

1. Done in Jupyter Notebook

2. In the CartPole problem (in the notebook) our state is the current position of the cart, the current velocity of the cart, the current (angular) position of the pole, and the (angular) speed of the pole. As these are continuous variables, we have an infinite number of states (ignoring the fact that a digital computer can only represent finitely

many states in finite memory). Can you think of a way in which we can still use a tabular approach? Can you think of an example problem where this would not work? Explain why would this work for CartPole and not for the example you mentioned.

> Yes, on the CartPole problem, which has continuous state variables, we can apply discretization. This involves dividing the range of each state variable into bins, effectively converting them into discrete states. For instance, we can divide cart position, velocity, pole angle, and pole angular speed into bins, making them manageable in a table.
>
> However, discretization may not work well for complex tasks like self-driving cars, where state variables, such as positions and velocities of various objects, resist easy discretization. This approach succeeds in CartPole because it has a simple environment and limited state variables which allows it. In more intricate scenarios, where small state changes have significant consequences, discretization can lead to information loss and suboptimal or unsafe actions.

Homework: REINFORCE

Consider an undiscounted Markov Decision Process (MDP) with two states A and B, each with two possible actions 1 and 2, and a terminal state T with $V(T) = 0$. The transition and reward functions are unknown, but you have observed the following two episodes using a policy $\pi(a|s,\theta)$. The form of π is not specified, however the parameters θ are split into θ_a and θ_b where the action in A only depends on θ_a and the action taken in state B only depends on θ_b ($\pi(a|A,\theta_a)$ and $\pi(a|B,\theta_b)$).

$$A \xrightarrow[r_1=200]{a_1=1} A \xrightarrow[r_2=-20]{a_2=2} B \xrightarrow[r_3=-2]{a_3=2} B \xrightarrow[r_4=-3]{a_4=2} T$$

$$A \xrightarrow[r_1=-100]{a_1=1} A \xrightarrow[r_2=20]{a_2=2} B \xrightarrow[r_3=10]{a_3=1} B \xrightarrow[r_4=10]{a_4=1} T$$

where the arrow ($\rightarrow$) indicates a transition and a_t and r_t take the values of the observed actions and rewards respectively.

1. For each episode, provide the expression for the update to parameters θ_a or θ_b given by the following algorithms:

(a) Classical REINFORCE

Given a trajectory (an episode) τ, the update to the policy parameters θ_a and θ_b is given by:

For state A (θ_a):

$$\theta_a^{t+1} = \theta_a + E[G(t) \sum_{t=1}^{t} \nabla_\theta log \pi_{\theta_a}(a_t|A)]$$

For state B (θ_b):

$$\theta_b^{t+1} = \theta_b + E[G(t) \sum_{t=1}^{t} \nabla_\theta log \pi_{\theta_b}(b_t|B)]$$

For the given episodes:

For the first episode:

Given the states and actions, the updates for θ_a and θ_b are:

$$\theta_a^{t+1} = \theta_a + 175 * (\nabla_\theta log \pi_{\theta_a}(1|A) + \nabla_\theta log \pi_{\theta_a}(2|A))$$
$$\theta_b^{t+1} = \theta_b + 175 * (\nabla_\theta log \pi_{\theta_b}(2|B) + \nabla_\theta log \pi_{\theta_b}(2|B))$$

For the second episode:

The updates for θ_a and θ_b are:

$$\theta_a^{t+1} = \theta_a - 60 * (\nabla_{\pi\theta_a}(1|A) + \nabla_\theta log \pi_{\theta_a}(2|A))$$
$$\theta_b^{t+1} = \theta_b - 60 * (\nabla_\theta log \pi_{\theta_b}(1|B) + \nabla_\theta log \pi_{\theta_b}(1|B))$$

(b) REINFORCE/G(PO)MDP

The update rule for the parameters θ is:

For state A (θ_a):

$$\theta_a^{t+1} = \theta_a + E[\sum_{t=1}^{T} \nabla_\theta log \pi_{\theta_a}(a_t|A) \sum_{t'=t+1:T} r_t']$$

For state B (θ_b):

$$\theta_b^{t+1} = \theta_b + E[\sum_{t=1}^{T} \nabla_\theta log \pi_{\theta_b}(b_t|B) \sum_{t'=t+1:T} r_t']$$

For the first episode:

Updates for θ_a and θ_b based on the states and actions:

For state A (θ_a):

$$\theta_a^{t+1} = \theta_a - 175 * (\nabla_\theta log\pi_{\theta_a}(1|A) - 25 * \nabla_\theta log\pi_{\theta_a}(2|A))$$

For state B (θ_b):

$$\theta_b^{t+1} = \theta_b - 5 * (\nabla_\theta log\pi_{\theta_a}(2|B) - 3\nabla_\theta log\pi_{\theta_b}(2|B))$$

For the second episode:

Updates for θ_a and θ_b based on the states and actions:
For state A (θ_a):

$$\theta_a^{t+1} = \theta_a + 60 * (\nabla_\theta log\pi_{\theta_a}(1|A) + 40 * \nabla_\theta log\pi_{\theta_a}(2|A))$$

For state B (θ_b):

$$\theta_b^{t+1} = \theta_b + 20 * (\nabla_\theta log\pi_{\theta_b}(1|B) + 10 * \nabla_\theta log\pi_{\theta_b}(1|B))$$

2. For each update, explain what impact you would expect the update to have on the probability of taking actions in state B.

Classical Reinforce

- episode 1: The probability of taking action 2 will go up because the update is positive
- episode 2: The probability of taking action 1 will go down because the update is negative

REINFORCE/G(PO)MDP

- episode 1: The probability of taking action 2 will go down because the update is negative
- episode 2: The probability of taking action 1 will go up because the update is positive

3. Which algorithm do you think leads to a better update? Provide an intuitive explanation as to why you think this is the case.

> We think that REINFORCE/G(PO)MDP leads to a better update. The gradient is positive for taking action 1 and negative for action 2. This corresponds to the rewards, which are positive for action 1 and negative for action 2. For classical Reinforce, these gradients are in opposite directions, which is counter-intuitive. Therefore, it makes sense that REINFORCE/G(PO)MDP provides a better update.

4. Consider policy gradient as approximated from a single episode using classical REINFORCE:

$$\nabla_\theta J(\theta) \approx (\sum_{k=0}^{T} R(S_k, A_k)) \sum_{t=0}^{T} \nabla_\theta log \pi_\theta(A_t|S_t)$$

and as approximated using REINFORCE/G(PO)MDP:

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{T} \nabla_\theta log \pi_\theta(A_t|S_t) \sum_{k=0}^{T} R(S_k, A_k)$$

Which of the two forms has less variance around the true value of the gradient? Please provide an explanation.

> – The classical REINFORCE method computes the sum of the log-probabilities of actions taken across the trajectory and multiplies this sum with the total reward obtained across the entire trajectory. This can introduce high variance since any deviation in reward or policy along the trajectory can significantly affect the estimated gradient.
>
> – The REINFORCE/G(PO)MDP method, on the other hand, associates each log-probability term with only the future cumulative reward from that time-step. This restricts the influence of a particular reward and policy to only the future cumulative rewards rather than the entire trajectory, which tends to reduce the variance.
>
> In conclusion, $\nabla_\theta J(\theta)$ as approximated using REINFORCE/G(PO)MDP is expected to have less variance around the true value of the gradient compared to the classical REINFORCE method. The reason is that REINFORCE/G(PO)MDP's

formulation provides a more direct and localized association between policy decisions and their consequences, thus making it less susceptible to the high variability that can arise when summing across the entire trajectory.

5. We can also combine policy and value based methods, suggest a variation of REINFORCE/G(PO)MDP which leverages a value function, and discuss the value function's role in this algorithm.

A variation of REINFORCE/G(PO)MDP that leverages a value function is known as the Actor-Critic algorithm. The central idea behind Actor-Critic is that while the policy (actor) is used to select actions, the value function (critic) provides a baseline to estimate the advantage of taking a specific action.

The role of the value function $V_{\theta_v}(S_t)$ in this algorithm is to reduce the variance. By subtracting the value function's estimate from the observed return, we are effectively using it as a baseline. This reduces the variance in our gradient estimates, leading to more stable training.