Exercise Set 4 - Reinforcement Learning

Control with approximation and policy gradients

Giulio Starace - 13010840

October 4, 2022

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

1. To compute the Bellman error vector after initialization, we compute the Bellman error for each state. We first recall the definition of the Bellman operator B^π:

$$(B^\pi v)(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v(s')\right]. \tag{1}$$

We can plug this into the definition of the Bellman error:

$$\begin{aligned}\overline{\delta}_w(s) &= B^\pi v_w - v_w \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_w(s')\right] - v_w(s).\end{aligned}$$

For s_0, we have a single action available that is always taken, so our $\sum_a \pi(a|s)$ term disappears and we are left with:

$$\overline{\delta}_w(s) = \sum_{s',r} p(s', r|s) \left[r + \gamma v_w(s')\right] - v_w(s) \tag{2}$$

Our action always leads to the same state, with the same reward (of 0), so we can simplify further and write

$$\overline{\delta}_w(s) = \gamma v_w(s') - v_w(s). \tag{3}$$

Finally, we have that $v_{w,s} = w \cdot \phi_s$, so that we can write

$$\overline{\delta}_w(s) = \gamma w \cdot \phi_{s'} - w \cdot \phi_s. \tag{4}$$

The same arguments can be applied to s_1, so we can write the Bellman error *vector* as

$$\text{BE}(w) = \left(\overline{\delta}_w(s_0),\ \overline{\delta}_w(s_1)\right)^T = \left(\gamma w \cdot \phi_{s_1} - w \cdot \phi_{s_0},\ \gamma w \cdot \phi_{s_0} - w \cdot \phi_{s_1}\right)^T. \tag{5}$$

We can plug in our values $w = 1$, $\phi_{s_0} = 2$, $\phi_{s_1} = 1$, and $\gamma = 1$ and obtain

$$\text{BE}(w) = \left(1 \cdot 1 - 1 \cdot 2,\ 1 \cdot 2 - 1 \cdot 1\right)^T = \left(-1,\ 1\right)^T. \tag{6}$$

2. The Mean Squared Bellman Error $\overline{\text{BE}}(w)$ is the measure of the overall error in the value function, computed by taking the μ weighted norm of the Bellman error vector. Here, μ is a distribution $\mu : \mathcal{S} \to [0, 1]$ specifying the extent to which each state is considered in the computation. Mathematically:

$$\overline{\text{BE}}(w) = \|\text{BE}(w)\|_\mu^2 = \sum_s \mu(s)\bar{\delta}_w(s)^2. \tag{7}$$

In our case, this would be expressed as

$$\overline{\text{BE}}(w) = \mu(s_0) \cdot (-1)^2 + \mu(s_1) \cdot 1^2 = \mu(s_0) + \mu(s_1) = \sum_s \mu(s). \tag{8}$$

3. The target values $B^\pi v_w$ we found in question 1. are 1 for s_0 and 2 for s_1. The w that results in the value function that is closest can be applied using a least-squares regression:

$$\begin{aligned} \beta &= \arg\min_\beta \|\mathbf{Y} - w\mathbf{X}\|^2 \\ w &= \arg\min_w \|(1,\ 2)^T - w(\phi_{s_0},\ \phi_{s_1})^T\|^2 \\ &= \arg\min_w \|(1,\ 2)^T - w(2, 1)^T\|^2. \end{aligned} \tag{9}$$

This has a closed-form solution:

$$\beta = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}, \tag{10}$$

which for our numbers gives $w = 4/5$.

4. The plot of v_w, $B^\pi v_w$ and $\Pi B^\pi v_w$ is shown in Figure 1. We see our value function v_w lying on the representable subspace that is the line (not plane, unlike Figure 11.3 in the book) defined by our scalar (not vector, unlike Figure 11.3 in the book) parameter w. We see that the application of the Bellman operator B^π to our value function v_w results in a new value function $B^\pi v_w$ outside our representable subspace. Like in Figure 11.3 in the book, we see that applying the projection operator Π to $B^\pi v_w$ then projects the value function back onto the representable subspace, as $\Pi B^\pi v_w$, which lies on w. In particular, $\Pi B^\pi v_w$ lies on the line at $w = 4/5$, since this is the value of w that minimizes the Bellman error.

Homework: Coding Assignment - Deep Q Networks

1. Coding answers have been submitted on codegra under the group "stalwart cocky sawly".

2. We can still use a tabular approach for the CartPole problem by discretizing the state space. We can do this by simply defining a grid of possible values for each of the state variables and grouping the continuous values into the appropriate bins. This approach is not possible for problems dealing with continuous control, particularly those with an infinite number of actions. An example of this training a robot hand to grasp a variety of objects without damage. The agent needs to learn to control the forces and position of the fingers, the former of which can be any real number, and may be sampled from a continuous distribution. In our CartPole example, we are able to discretize both the state space and action space since we have well defined limits on both what the state variables can be (e.g. a degree between -24 and 24) and what actions we can take (e.g. go left or go right).

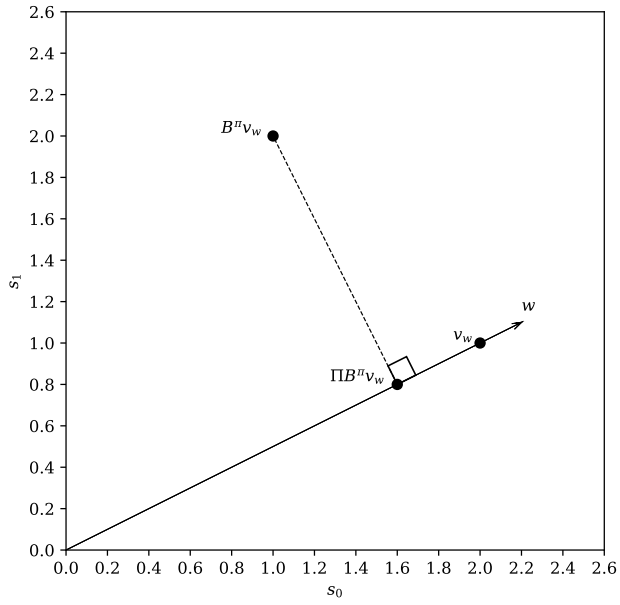

Figure 1: Plot of v_w, $B^\pi v_w$ and $\Pi B^\pi v_w$

Homework: REINFORCE

1. (a) The update to the policy parameter θ under classical REINFORCE after the eth episode is given by

$$\theta_{e+1} \leftarrow \theta_e + \alpha \widehat{\nabla_\theta J}, \tag{11}$$

where $\widehat{\nabla J}$ is an estimate of the gradient the expectation of the return $G(\tau_e)$ for a given trajectory τ, and α is the learning rate. This is given by

$$\widehat{\nabla_\theta J} = \frac{1}{N} \sum_{i=1}^{N} \left[G(\tau_i) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \right], \tag{12}$$

where N is the number of trajectories observed, and π_θ is the policy we are parametrising with θ. Because we are interested in the update per-episode, $N = 1$ and our equation simplifies to

$$\widehat{\nabla_\theta J} = G(\tau_e) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \tag{13}$$

Because in our case θ is split into θ_a and θ_b, where an action in state A only depends on θ_a and the action in state B only depends on θ_b, we can treat the update to each parameter separately, obtaining

$$\theta_a^{e+1} \leftarrow \theta_a^e + \alpha \widehat{\nabla_{\theta_a} J} \tag{14}$$

$$\theta_b^{e+1} \leftarrow \theta_b^e + \alpha \widehat{\nabla_{\theta_b} J}. \tag{15}$$

$\widehat{\nabla_{\theta_{a/b}} J}$ takes the same form as (13), but now only depends on the actions taken in state A or B respectively. This leaves us with

$$\widehat{\nabla_{\theta_a} J} = G(\tau_e) \sum_{t:s_t=A}^{T} \nabla_{\theta_a} \log \pi_\theta(a_t^i | A, \theta_a) \tag{16}$$

$$\widehat{\nabla_{\theta_b} J} = G(\tau_e) \sum_{t:s_t=B}^{T} \nabla_{\theta_b} \log \pi_\theta(a_t^i | B, \theta_b) \tag{17}$$

We can plug this into our updates for each part of θ and obtain

$$\theta_a^{e+1} \leftarrow \theta_a^e + \alpha G(\tau_e) \sum_{t:s_t=A}^{T} \nabla_{\theta_a} \log \pi_\theta(a_t^i | A, \theta_a) \tag{18}$$

$$\theta_b^{e+1} \leftarrow \theta_b^e + \alpha G(\tau_e) \sum_{t:s_t=B}^{T} \nabla_{\theta_b} \log \pi_\theta(a_t^i | B, \theta_b), \tag{19}$$

where the trajectory τ_e will be different for each episode e. We can plug in the values given from our sampled episodes and obtain the following four updates:

$$\theta_a^1 \leftarrow \theta_a^0 + \alpha \cdot 175 \cdot \nabla_{\theta_a} \left[\log \pi_\theta(1|A, \theta_a) + \log \pi_\theta(2|A, \theta_a)\right] \tag{20}$$

$$\theta_a^2 \leftarrow \theta_a^1 - \alpha \cdot 60 \cdot \nabla_{\theta_a} \left[\log \pi_\theta(1|A, \theta_a) + \log \pi_\theta(2|A, \theta_a)\right] \tag{21}$$

$$\theta_b^1 \leftarrow \theta_b^0 + \alpha \cdot 350 \cdot \nabla_{\theta_b} \log \pi_\theta(2|B, \theta_b) \tag{22}$$

$$\theta_b^1 \leftarrow \theta_b^0 - \alpha \cdot 120 \cdot \nabla_{\theta_b} \log \pi_\theta(1|B, \theta_b) \tag{23}$$

(b) When using REINFORCE/G(MO)MDP, we now have a different general estimate of the gradient of the expectation of the return, $\widehat{\nabla J}$, given by

$$\widehat{\nabla_\theta J} = \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} r_t \sum_{t'=1}^{t} \nabla_\theta \log \pi_\theta(a_{t'} | s_{t'}) \right), \tag{24}$$

where r_t is the reward at time t for episode i. We can apply the same reasoning used in 1a and obtain the update rule per episode for each split of θ:

$$\theta_a^{e+1} \leftarrow \theta_a^e + \alpha \sum_{t=1}^{T} r_t \sum_{t' \in \{z:s_z=A \wedge z \leq t\}} \nabla_{\theta_a} \log \pi_\theta(a_{t'} | A, \theta_a) \tag{25}$$

$$\theta_b^{e+1} \leftarrow \theta_b^e + \alpha \sum_{t=1}^{T} r_t \sum_{t' \in \{z:s_z=B \wedge z \leq t\}} \nabla_{\theta_b} \log \pi_\theta(a_{t'} | B, \theta_b) \tag{26}$$

where the rewards r_t will be different depending on the episode. We can plug in the values given from our sampled episodes and obtain the following four updates:

$$\theta_a^1 \leftarrow \theta_a^0 + \alpha \cdot \nabla_{\theta_a} \left[175 \cdot \log \pi(1|A, \theta_a) - 25 \cdot \log \pi(2|A, \theta_a)\right] \tag{27}$$

$$\theta_a^2 \leftarrow \theta_a^1 + \alpha \cdot \nabla_{\theta_a} \left[-60 \cdot \log \pi(1|A, \theta_a) + 40 \cdot \log \pi(2|A, \theta_a)\right] \tag{28}$$

$$\theta_b^1 \leftarrow \theta_b^0 + \alpha \cdot \nabla_{\theta_b} \left[-8 \cdot \log \pi(2|B, \theta_B)\right] \tag{29}$$

$$\theta_b^2 \leftarrow \theta_b^1 + \alpha \cdot \nabla_{\theta_b} \left[30 \cdot \log \pi(1|B, \theta_B)\right] \tag{30}$$

2. hello world

3. G(PO)MDP will lead to a better update than Classical REINFORCE because the latter has higher variance in its estimates. When using the same learning rate, this makes it more likely for Classical REINFORCE to overshoot the optimal update iteration.

4. Between the approximated policy gradient for a single episode using classical REINFORCE and that using REINFORCE/G(PO)MDP, the latter has less variance around the true value of the gradient. Intuitively, this is because in G(PO)MDP we only consider relevant rewards, as opposed to classical REINFORCE where we consider the return for the whole episode. This leads to less accumulation of variance from the sampled rewards.

5. A variation of G(PO)MDP which leverages the value function can be devised by subtracting an estimate of the value function from the return at each time step. Here the value function serves as a baseline for the reward, so to reduce variance in the gradient estimate.