NAME1: Philipp Lintl
STUDENTID1:12152498
MAIL1: philipp.lintl@student.uva.nl

NAME2: Bogdan Floris
STUDENTID2: 12140910
MAIL2 bogdan.floris@student.uva.nl

Homework Assignment 1

Reinforcement Learning, 19/20

2019-10-04

5.1 Basis functions

1. Tabular methods map discrete states to values. A linear function approximation can be seen as a tabular method, when using a one-hot encoding for the states. This leads to a transformation/feature vector x, that only contains one 1, which is on the position of the state that is to be evaluated. So the linear approximation $\hat{v}(s, \mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s_j) \doteq \sum_{i=1}^{d} w_i x_i(s_j)$ gets $\sum_{i=1}^{d} w_i x_j(j) = w_j \cdot x_j(s_j) = w_j \cdot s_j$, as the transformation is identity.

2. Given $s = [x, y]$. Interaction among them. Good polinomial feature vector? To account for interactions, we need $x(s) = (x, y, x \cdot y)$ or even higher polynomials $(e.g.(x, y, xy, x^2, y^2$. If prior knowledge is available, the most important feature combinations can be chosen.

3. Increases exponentially for linear increase in number of states.

4. For polynomial basis functions, we could add more dimensions that entail the feature to be evaluated more important. Weight it higher: e.g. for radial basis functions: allow skew in the axis of the state space that it to be regarded more important, so the receiptive field is more variant in that dimensions.

5. If the presence of a feature in coarse coding is modeled binarily and not assessing a degree of presence ([0,1]), this is the same as taking a threshold value for RBFs and then evaluate the RBF response with regards to the threshold.

5.2 Neural Networks

1. Neural networks in this scenario are used to minimize an error between the approximated value function and the true value function. For instance looking at the VE (mean squared value error), the state distribution $\mu(s)$ weights the difference between the quadratic difference between the approximated and the true value function. This ensures that frequently occuring states are assessed more accurately than seldomly occuring states. The learned policy is changed at the update step of the approximation functions parameters. This policy change then in turn changes the state distribution. After convergence of the weight update, the state distribution will be converged as well making it a stationary distribution.

2. In reinforcement learning, updates are changed online (while looping through the data, in this case episodes). In standard supervised learning, the entire dataset is passed once before changing parameters. Also, supervised learning methods usually assume independent observations, which is not the case in reinforcement learning, where an episode consists of subsequent, dependent observations.
   In supervised learning the chosen model is the one minimizing the loss function the best, whereas in reinforcement learning this does not hold. In RL we are interested in the optimal policy, not the optimal value function.

3. In normal supervised Learning errors are weighted equally across all observations. However, when for instance class imbalance is observed, one strategy to overcome this problem is to assign weights to classes. That way, infrequent classes can be weighted higher in order to balance them being less frequent. Also, if some classes are priorly known to be more important, they can be weighted accordingly.
   In reinforcement learning, the weighting is based on the frequency of the state itself and thus, contrary to the supervised approach, not known prior to the estimation process but obtained during.

4. As mentioned, RL does not fulfil the independence assumption of observations, that NN usually rely on. This stems from the fact that in episodic RL, observations are formed from state, action, next state, reward tuples. Thus, observations within one episode are highly dependent. In order to break the dependence among observations of an episode, each is stored in a memory buffer. This buffer is then subject to random selection of a batch. The weights of an estimation or control algorithm are updated based on these sampled experiences, whereas the simulation of steps is continued after the training step.

5. When the target network is 'frozen' for periods of time, the estimate and target are decorrelated. So, after a fixed number of steps, the target network is updated with the current estimate again. Thus, not only decorrelation, but also higher stability is achieved. When the current network yields a large weight update, but the target network is not the same, the weight updates will not be as large.

5.4 HOMEWORK: Gradient Descent Methods

1. Given, interaction with the environment generates states in accordance to a policy π. Since the true value of a state is the expected value of the subsequent return, by-definition the Monte Carlo target is an unbiased estimate of $v_\pi(S_t)$.
2. As seen in the given formula, Bootstrapping targets, such as Dynamic Programming target, depend on the current value of the weight vector $\mathbf{w}_t$. Thus, the the targets are biased and both targets and estimates depend on the function approximation of the value function. Therefore, a true gradient descent, which requires unbiased estimates is not given. To overcome this, the gradient is only taken with regards to the estimate, also known as the semi-gradient method.
3. As discussed, Bootstrapping methods are biased and therefore do not converge as robustly as MC methods. However, using the bootstrapped estimate of future rewards rather than the actual return observed in Monte Carlo, speeds up the learning significantly. Accordingly, the policy is updated at each time step the value function is updated.

   A characteristic to the Mountain Car problem is that rewards are only rarely registered, as rewards are only collected after termination. An episode only terminates, if the car reaches the top. In order to end up at the top, a complex sequence of states and actions is required, which leads to very few termination observations. Therefore, the online learning possibility within bootstrapping methods speeds up the learning process.

6.1 HOMEWORK: Geometry of linear value-function approximation

1. Given the Bellman error vector: $\bar{\delta}_w = B_\pi v_w - v_w$
   The Bellman operator is defined as

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

   , which due to deterministic character and only one possible next action in this scenario comes down the term in the last bracket:

$$(B_\pi v)(s) = \left[r + \gamma v(s')\right]$$

   Taking $w = 1$ and $\gamma = 1$, the initial state value approximation looks as such:

$$v_\pi(\mathbf{s}) = w \cdot \phi = 1 \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

   Plugging this into the Bellman error vector:

$$\bar{\delta}_w = B_\pi v_w - v_w = \begin{bmatrix} r_1 + \gamma v_w(s_2) - v_w(s_1) \\ r_2 + \gamma v_w(s_1) - v_w(s_2) \end{bmatrix} = \begin{bmatrix} 0+2-1 \\ 0+1-2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

2. The mean squared Bellman error is defined as such $\overline{\mathrm{BE}}(\mathbf{w}) = \|\bar{\delta}_\mathbf{w}\|_\mu^2$, with the norm being 11.11 from Sutton and Barto: $\|v\|_\mu^2 \doteq \sum_{s\in\mathcal{S}} \mu(s)v(s)^2$
   Both states appear equally often, therefore, the state distribution $\mu(s) = \frac{1}{2}$ for both of them. Then:

$$\overline{\mathrm{BE}}(\mathbf{w}) = \frac{1}{2}(1^2 + (-1)^2) = 1$$

3. We want identify the weight parameters, that result in the value function closest to the Bellman operator of v_w. Taking the initialization, we see the first Bellman operator step as:

$$B_\pi v_w = \begin{bmatrix} r_1 + \gamma v_w(s_2) \\ r_2 + \gamma v_w(s_1) \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Following Sutton and Bartos 11.12, we end up with $w^\star$ of $v_{w^\star}$ that is closest to B_piv_w by applying the projection operator:

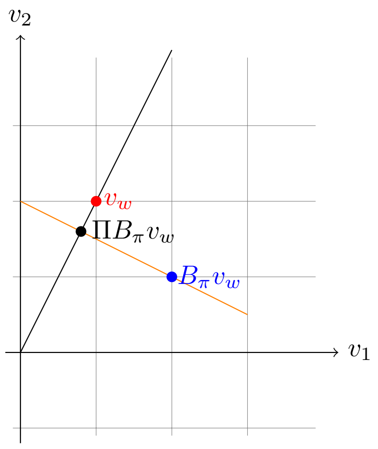
$$v_{w*} = \Pi B_\pi v_w$$

The respective $w^\star$ is obtained by

$$w^\star = \arg\min_{w^\star} \|B_\pi v_w - v_{w^\star}\|^2_\mu$$

Deriving wrt $w^\star$ and setting 0:

$$\begin{aligned}
0 &= \frac{\partial \|B^\pi v_w - v_{w*}\|^2_\mu}{\partial w^*} \\
0 &= \frac{\partial \frac{1}{2}\left((2-w^*)^2 + (1-2w^*)^2\right)}{\partial w^*} \\
0 &= -4 + 2w^\star - 4 + 8w^\star \\
0 &= 10w^* - 8 \\
w^* &= \frac{4}{5} \\
\Rightarrow v_{w*} = \Pi B_\pi v_w &= \begin{bmatrix} \frac{4}{5} \\ \frac{8}{5} \end{bmatrix}
\end{aligned}$$

4. 

We start with v_w for parameter $w = 1$. By applying the Bellman operator to v_w, we end up at the target value $B_\pi v_w$ (blue point). The black line in the plot illustrates the representable 1-dimensional manifold of state values for a changing w, an the orange line marks the respective representation in the original 2-dimensional representation. The projection operator Π projects the Bellman target back into the 1-dimensional manifold. We found the respective $w^\star$ by minimizing the distance between b_piv_w and v_w. The black line is also referred to as function approximation manifold, as this is the subspace, in which the approximation takes place.

7.1 REINFORCE

1. Using $Var(x) = E(X^2) - E(X^2)$:

$$\begin{aligned}
\mathbb{V}\left[(G_\tau - b)\nabla \log p(\tau)\right] &= \mathbb{E}_\tau\left[(G_\tau - b)^2 (\nabla \log p(\tau))^2\right] - \left(\mathbb{E}_\tau\left[(G_\tau - b)\nabla \log p(\tau)\right]\right)^2 \\
&= \mathbb{E}_\tau\left[\left(G_\tau^2 - 2bG_\tau + b^2\right)(\nabla \log p(\tau))^2\right] - \left(\mathbb{E}_\tau\left[(G_\tau - b)\nabla \log p(\tau)\right]\right)^2 \\
&= \mathbb{E}_\tau\left[G_\tau^2(\nabla \log p(\tau))^2\right] - \left(\mathbb{E}_\tau\left[(G_\tau - b)\nabla \log p(\tau)\right]\right)^2 - 2b\mathbb{E}_\tau\left[G_\tau(\nabla \log p(\tau))^2\right] \\
&+ b^2\mathbb{E}_\tau\left[(\nabla \log p(\tau))^2\right] \\
&= \mathbb{V}_\tau\left[G_\tau(\nabla \log p(\tau))\right] - 2b\mathbb{E}_{\tau\tau}\left[G_\tau(\nabla \log p(\tau))^2\right] + b^2\mathbb{E}_\tau\left[(\nabla \log p(\tau))^2\right]
\end{aligned}$$

We need to derive this term for b, in order to end up with a baseline value b, that minimizes the variance term:

$$\begin{aligned}\frac{\partial \mathbb{V}_\tau\left[(G_\tau - b)\nabla \log p(\tau)\right]}{\partial b} &= -2\mathbb{E}_\tau\left[G_\tau(\nabla \log p(\tau))^2\right] + 2b\mathbb{E}_\tau\left[(\nabla \log p(\tau))^2\right] = 0 \\ 2b\mathbb{E}_\tau\left[(\nabla \log p(\tau))^2\right] &= 2\mathbb{E}_\tau\left[G_\tau(\nabla \log p(\tau))^2\right] \\ \Rightarrow b &= \frac{\mathbb{E}_\tau\left[G_\tau(\nabla \log p(\tau))^2\right]}{\mathbb{E}_\tau\left[(\nabla \log p(\tau))^2\right]}\end{aligned}$$

2. Given: r=a+2; $a \sim \mathcal{N}(\theta, 1)$; $\nabla_\theta \log(\pi(a)) = a - \theta$ One trajectory τ only consists of one action a and thus the return is only one reward $\Rightarrow G(\tau) = G(a) = a + 2$. The probability of the trajectory therefore also transforms to $p(\tau) = p(a)$.

   Plugging those into the derived optimal b value:

$$\begin{aligned} &= \frac{\mathbb{E}_a\left[G(a)\nabla \log p(a)^2\right]}{\mathbb{E}_a\left[\nabla \log p(a)^2\right]} = \frac{\mathbb{E}_a\left[(a+2)(a-\theta)^2\right]}{\mathbb{E}_a\left[(a-\theta)^2\right]} \\ &= \frac{\mathbb{E}_a\left[a^3 - 2a^2\theta + a\theta^2 + 2a^2 - 4a\theta + 2\theta^2\right]}{\mathbb{E}_a\left[a^2 - 2a\theta + \theta^2\right]} \\ &= \frac{\mathbb{E}_a\left[a^3\right] - 2\mathbb{E}_a[a^2]\theta + \mathbb{E}_a[a]\theta^2 + 2\mathbb{E}_a\left[a^2\right] - 4\mathbb{E}_a[a]\theta + 2\theta^2}{\mathbb{E}_a\left[a^2 - 2a\theta + \theta^2\right]} \\ &= \frac{\mathbb{E}_a\left[a^3\right] + 2(1-\theta)\mathbb{E}_a\left[a^2\right] + \theta^3 - 4\theta^2 + 2\theta^2}{(1+\theta^2) - 2\theta^2 + \theta^2} \\ &= \frac{(\theta^3 + 3\theta) + 2(1-\theta)(1+\theta^2) + \theta^3 - 4\theta^2 + 2\theta^2}{(1+\theta^2) - 2\theta^2 + \theta^2} \\ &= \frac{(\theta^3 + 3\theta) + 2 + 2\theta^2 - 2\theta - 2\theta^3 + \theta^3 - 4\theta^2 + 2\theta^2}{1} = \theta + 2 \end{aligned}$$

   For $\mathbb{E}[a^2]$ and $\mathbb{E}[a^3]$, we use the information that a is Gaussian $\mathcal{N}(\theta, 1))$ distributed. We take advantage of the second and third moment being:

$$\mathbb{E}[a^2] = \mu^2 + \sigma^2 = \theta^2 + 1 \qquad\qquad \mathbb{E}[a^3] = \mu^3 + 3\mu\sigma^2 = \theta^3 + 3\theta$$

8.1 Limits of policy gradients

8.2 HOMEWORK: Compatible Function Approximation Theorem

1.

$$\begin{aligned}\mathbb{E}\left[\hat{q}_w(s,a)\right] &= \mathop{\mathbb{E}}_a\left[w^T \nabla_\theta \log \pi_\theta(a,s)\right] \\ &= w^T \sum_a \pi_\theta(a,s)\nabla_\theta \log \pi_\theta(a,s) \\ &= w^T \sum_a \pi_\theta(a,s)\frac{\nabla_\theta \pi_\theta(a,s)}{\pi_\theta(a,s)} \\ &= w^T \sum_a \nabla_\theta \pi_\theta(a,s) \\ &= w^T \nabla_\theta \left(\sum_a \pi_\theta(a,s)\right) \\ &= w^T \nabla_\theta 1 = w^T 0 = 0\end{aligned}$$

   The expectation over all actions over the approximated q function $\hat{q}_w(s,a)$ is zero. This means that is not an unbiased estimate of the real q function, which otherwise would yield an expectation of q_π. This is reasonable, as the compatible function approximation theorem has to be satisfied. This implies that the average gradients obtained by $\hat{q}(s,a)$ is equal to the average obtained by using the true action-value function. This ensures that the actor receives correct gradients.

2.

$$\begin{aligned}\mathbb{E}\left[q_\pi(s,a) - v_\pi(s),\right] &= \mathbb{E}\left[q_\pi(s,a)\right] - \mathbb{E}\left[v_\pi(s)\right] \\ &= v_\pi(s) - v_\pi(s) \\ &= 0\end{aligned}$$

3. As the expectation of our approximation $\hat{q}(s,a)$ is equal to the expectation of the advantage function, optimizing our approximation can be interpreted like optimizing the advantage function.

4.

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a,s) &= \frac{1}{\pi_\theta(a,s)} \nabla_\theta \pi_\theta(a,s) \\
&= \frac{\sum_b e^{\theta^T \phi_{sb}}}{e^{\theta^T \phi_{sa}}} \cdot \nabla_\theta \frac{e^{\theta^T \phi_{sa}}}{\sum_b e^{\theta^T \phi_{sb}}} \\
=& \frac{\sum_b e^{\theta^T \phi_{sb}}}{e^{\theta^T \phi_{sa}}} \cdot \frac{\left(\nabla_\theta e^{\theta^T \phi_{sa}}\right) \sum_b e^{\theta^T \phi_{sb}} - e^{\theta^T \phi_{sa}} \nabla_\theta \sum_b e^{\theta^T \phi_{sb}}}{\left(\sum_b e^{\theta^T \phi_{sb}}\right)^2} \\
=& \frac{1}{e^{\theta^T \phi_{sa}}} \frac{\phi_{sa} e^{\theta^T \phi_{sa}} \sum_b e^{\theta^T \phi_{sb}} - e^{\theta^T \phi_{sa}} \sum_b \phi_{sb} e^{\theta^T \phi_{sb}}}{\sum_b e^{\theta^T \phi_{sb}}} \\
&= \frac{\phi_{sa} \sum_b e^{\theta^T \phi_{sb}} - \sum_b \phi_{sb} e^{\theta^T \phi_{sb}}}{\sum_b e^{\theta^T \phi_{sb}}} \\
&= \phi_{sa} - \sum_b \phi_{sb} \frac{e^{\theta^T \phi_{sb}}}{\sum_b e^{\theta^T \phi_{sb}}} \\
&= \phi_{sa} - \sum_b \pi(s,b) \phi_{sb}
\end{aligned}$$

That is, why we can express $\hat{q}$ as

$$\begin{aligned}
\hat{q}_w &= w^T \nabla_\theta \log \pi_\theta(s,a) \\
&= w^T \left(\phi_{sa} - \sum_b \pi(s,b) \phi_{sb} \right)
\end{aligned}$$