Solutions for week 4
Reinforcement Learning: Basics

Exercise 1: Iterative update[1]

We consider an empirical evaluation of $Q(s,a)$ by averaging the rewards for action a over the first k trials:

$$Q_k = \frac{1}{k}\sum_{i=1}^{k} r_i.$$

We now include an additional trial and average over all $k+1$ trials.

a. Show that this procedure leads to an iterative update rule of the form

$$\Delta Q_k = \eta_k(r_k - Q_{k-1}),$$

(assuming $Q_0 = 0$).

b. What is the value of η_k?

c. Give an intuitive explanation of the update rule.
Hint: Think of the following: If the actual reward is larger than my estimate, then I should ...

Solution:

a. We define ΔQ_k as the difference between Q_k and Q_{k-1}, and we simplify:

$$\begin{aligned}
\Delta Q_k = Q_k - Q_{k-1} &= \frac{1}{k}\sum_{i=1}^{k} r_i - \frac{1}{k-1}\sum_{i=1}^{k-1} r_i \\
&= \frac{1}{k}\left(r_k + \sum_{i=1}^{k-1} r_i\right) - \frac{1}{k-1}\sum_{i=1}^{k-1} r_i \\
&= \frac{1}{k}\left(r_k + \frac{k-1}{k-1}\sum_{i=1}^{k-1} r_i - \frac{k}{k-1}\sum_{i=1}^{k-1} r_i\right) \\
&= \frac{1}{k}\left(r_k - \frac{1}{k-1}\sum_{i=1}^{k-1} r_i\right) \\
&= \eta_k \left(r_k - Q_{k-1}\right) .
\end{aligned}$$

b. $\eta_k = 1/k$.

c. If the actual reward is larger than my estimate, then I should increase my estimate, otherwise I should decrease it.

Exercise 2: Greedy policy and the two-armed bandit

In the "2-armed bandit" problem, one has to choose one of 2 actions. Assume action a_1 yields a reward of $r = 1$ with probability $p = 0.25$ and 0 otherwise. If you take action a_2, you will receive a reward of $r = 0.4$ with probability $p = 0.75$ and 0 otherwise. The "2-armed bandit" game is played several times and Q values are updated using the update rule $\Delta Q(s,a) = \eta[r_t - Q(s,a)]$.

[1] The result is also used in class; the calculation is analog to a calculation that was done for online k-means clustering.

a. Assume that you initialize all Q values at zero. You first try both actions: in trial 1 you choose a_1 and get $r = 1$; in trial 2 you choose a_2 and get $r = 0.4$. Update your Q values ($\eta = 0.2$).

b. In trials 3 to 5, you play greedy and always choose the action which looks best (i.e., has the highest Q-value). Which action has the higher Q-value after trial 5? (Assume that the actual reward is $r = 0$ in trials 3-5.)

c. Calculate the expected reward for both actions. Which one is the best?

d. Initialize both Q-values at 2 (optimistic). Assume that, as in the first part, in the first two trials you get for both actions the reward. Update your Q values once with $\eta = 0.2$. Suppose now that in the following rounds, in order to explore well, you choose actions a_1 and a_2 alternatingly and update the Q-values with a very small learning rate ($\eta = 0.001$). How many rounds (one round = two trials = one trial with each action) does it take *on average*, until the maximal Q-value also reflects the best action?
Hint: For $\eta \ll 1$ we can approximate the actual returns r_t with their expectations $E[r]$.

Solution:

a. In the beginning, $Q(a_1, t = 0) = Q(a_2, t = 0) = 0$ (we dropped the state index s since there is only a single state). After choosing action a_1 and receiving a reward of $r = 1$, its Q-value is updated to:

$$Q(a_1, t = 1) = Q(a_1, t = 0) + \Delta Q(a_1) = 0 + \eta(r - Q(a_1, t = 0)) = 0 + 0.2 \cdot 1 = 0.2.$$

After choosing action a_2 and receiving a reward of $r = 0.4$, its Q-value is updated to:

$$Q(a_2, t = 2) = Q(a_2, t = 0) + \Delta Q(a_2) = 0 + \eta(r - Q(a_2, t = 0)) = 0 + 0.2 \cdot 0.4 = 0.08.$$

Continuing with a greedy method implies that in the next round, action a_1 will be chosen.

b. In trial 3 you take action a_1. If the return is 0,

$$Q(a_1, t = 3) = Q(a_1, t = 2) + \eta(r - Q(a_1, t = 2)) = (1-\eta) \cdot Q(a_1, t = 2) + \eta r = 0.8 \cdot 0.2 = 0.16\,.$$

Thus, in trial 4 we take again action a_1. If the return is again 0,

$$Q(a_1, t = 4) = (1 - \eta) \cdot Q(a_1, t = 3) = 0.8 \cdot 0.16 = 0.128\,.$$

In trial 5 we take again action a_1. If the return is again 0,

$$Q(a_1, t = 5) = (1 - \eta) \cdot Q(a_1, t = 4) = 0.8 \cdot 0.128 = 0.1024\,.$$

Thus, with a greedy policy, also in trial 6 action a_1 will be taken. If by chance some of the returns were 1 instead of 0, $Q(a_1, t = 5)$ would be even higher, while $Q(a_2, t = 5) = Q(a_2, t = 2) = 0.08$ because action a_2 was never taken.

c. For action a_1, the expected reward per round is given by $E[r_1] = p \cdot 1 + (1-p) \cdot 0 = 0.25$. For action a_2, the expected reward per round is evaluated to $E[r_2] = 0.75 \cdot 0.4 + 0.25 \cdot 0 = 0.3$. The second action yields a higher reward on average.

d. Similarly as in a, we can compute the Q-values after the first step with $\eta = 0.2$. We obtain: $Q^*(a_1) = 1.8$ and $Q^*(a_2) = 1.68$. We use the hint that for $\eta \ll 1$ we can approximate the actual returns r_t with their expectations $E[r]$, i.e.

$$\begin{aligned} Q(a_i, t) &= (1 - \eta) Q(a_i, t - 1) + \eta r_t && (1)\\ &\approx (1 - \eta) Q(a_i, t - 1) + \eta E[r] && (2)\\ &= (1 - \eta)\left[(1 - \eta) Q(a_i, t - 2) + \eta E[r]\right] + \eta E[r] && (3) \end{aligned}$$

We continue by induction and arrive at

$$= \eta \sum_{s=0}^{t-1} (1-\eta)^s E[r] + (1-\eta)^t Q(a_i, 0) \tag{4}$$

$$= \eta \frac{1-(1-\eta)^t}{\eta} E[r] + (1-\eta)^t Q(a_i, 0) \tag{5}$$

$$= (1-\gamma^t) E[r] + \gamma^t Q(a_i, 0), \tag{6}$$

with $\gamma = 1 - \eta$ and using the formula for the geometric series. We search for the smallest t such that

$$Q(a_2, t) > Q(a_1, t) \tag{7}$$

$$(1-\gamma^t) E[r_2] + \gamma^t Q^*(a_2) > (1-\gamma^t) E[r_1] + \gamma^t Q^*(a_1) \tag{8}$$

$$\Rightarrow \gamma^t \left(Q^*(a_1) - Q^*(a_2) + E[r_2] - E[r_1] \right) < E[r_2] - E[r_1] \tag{9}$$

$$\Rightarrow t \log(\gamma) < \log \left(\frac{E[r_2] - E[r_1]}{Q^*(a_1) - Q^*(a_2) + E[r_2] - E[r_1]} \right) \tag{10}$$

$$\Rightarrow t > 1223.16 \tag{11}$$

Exercise 3: Batch vesrsus online learning rules: Recap

We define the mean squared error in a dataset with P data points as

$$E^{\text{MSE}}(\boldsymbol{w}) = \frac{1}{2} \frac{1}{P} \sum_{\mu} (t^\mu - \hat{y}^\mu)^2 \tag{12}$$

where the output is

$$\hat{y}^\mu = g(a^\mu) = g(\boldsymbol{w}^T \boldsymbol{x}^\mu) = g(\sum_k w_k x_k^\mu) \tag{13}$$

and the input is the $\boldsymbol{x}^\mu$ with components $x_1^\mu \dots x_d^\mu$.

a. Calculate the update of weight w_j by gradient descent (batch rule)

$$\Delta w_j = -\eta \frac{dE}{dw_j} \tag{14}$$

Hint: Apply chain rule

b. Rewrite the formula by taking one data point at a time (stochastic gradient descent). What is the difference to the batch rule?

c. Rewrite your result in b in vector notation (hint: use the weight vector $\boldsymbol{w}$ and the input vector $\boldsymbol{x}^\mu$). Show that the update after application of data point μ can be written as

$$\Delta \boldsymbol{w} = \eta \delta(\mu) \boldsymbol{x}^\mu$$

where $\delta(\mu)$ is a scalar number that depends on μ. Express $\delta(\mu)$ in terms of $t^\mu, \hat{y}^\mu, g'$.

d. The result is a non-Hebbian rule. Nevertheless, please try to identify the 'pre' contribution and the 'post' contributions. Which term makes it non-Hebbian?

Solution:

a.

$$\begin{aligned}\frac{dE}{dw_j} &= \frac{d}{dw_j}\frac{1}{2}\frac{1}{P}\sum_\mu (t^\mu - g(\sum_k w_k x_k^\mu))^2 \\ &= \frac{1}{2}\frac{1}{P}\sum_\mu 2(t^\mu - \hat{y}^\mu)\frac{d}{dw_j}\left(t^\mu - g(\sum_k w_k x_k^\mu)\right) \\ &= \frac{1}{P}\sum_\mu (t^\mu - g(a^\mu))(-x_j^\mu)g'(a^\mu)\end{aligned}$$

hence

$$\Delta w_j = \eta\,\frac{1}{P}\sum_\mu (t^\mu - g(a^\mu))x_j^\mu g'(a^\mu)$$

b.

$$\Delta w_j(\mu) = \eta\,(t^\mu - g(a^\mu))g'(a^\mu)x_j^\mu$$

Here, the weight update is performed on the error signal of a single data point. While in (a), the error signal is averaged over all data points. From a geometric perspective, the batch rule updates the separation hyperplane in the direction so that the hyperplane linearly separates all points. On the other hand, the stochastic *online* rule, will update the hyperplane so that a single randomly chosen point is pushed on the correct side of the hyperplane. Looking at the hyperplane updates, the stochastic rule leads to more noisy updates (the hyperplane moves in many different direction depending on the selected example) than the batch rule.

c. We know that $\boldsymbol{w} = [w_1, \ldots, w_d]$ where d is the dimensionality of the input space. $\Delta\boldsymbol{w} = [\Delta w_1, \ldots, \Delta w_d]$ and therefore the update rule in b in the vector form can be written as

$$\Delta\boldsymbol{w} = \eta\,\left[(t^\mu - g(a^\mu))g'(a^\mu)x_1^\mu, \ldots, (t^\mu - g(a^\mu))g'(a^\mu)x_d^\mu\right].$$

Taking the common scalar terms outside and substituting $g(a^\mu)$ with $\hat{y}^\mu$, we have

$$\Delta\boldsymbol{w} = \eta\,(t^\mu - \hat{y}^\mu)g'(a^\mu)\boldsymbol{x}^\mu$$

and thus $\delta(\mu) = (t^\mu - \hat{y}^\mu)g'(a^\mu)$.

d. Here $\boldsymbol{x}^\mu$ and $\hat{y}^\mu$ are the usual pre- and post-synaptic activities respectively. $g'(a^\mu)$ is the derivative of the postsynaptic activity and can be considered a postsynaptic term as well. It is the term t^μ, which is an external target value not dependent in the pre- or post-synaptic activities, that makes this update non-Hebbian.

Exercise 4: Geometric interpretation of an artificial neuron: Recap

Consider the single-neuron function in 2-D with

$$y = g(\boldsymbol{x}^T\boldsymbol{w}) \tag{15}$$

where g is a strictly increasing activation function, $\boldsymbol{x} = (x_1, x_2, -1) \in \mathbb{R}^{2+1}$ is the extended 2-dimensional input (i.e., the threshold/bias value has been integrated as an extra input $x_3 = -1$), and $\boldsymbol{w} = (w_1, w_2, w_3) \in \mathbb{R}^3$ is the weight vector. The hyperplane $\boldsymbol{x}^T\boldsymbol{w} = 0$ describes the boundary between where the neuron is on, i.e., $\boldsymbol{x}^T\boldsymbol{w} > 0$, and where it is off, i.e., $\boldsymbol{x}^T\boldsymbol{w} < 0$. Consider this hyperplane in the 2-D space of (x_1, x_2) and answer the following questions:

a. The hyperplane is a line in 2-D. What is the slope of this line as a function of w_1, w_2, and w_3? Where does the line intersect with the y-axis and where with the x-axis?

b. Is it possible to have two weight vectors $\boldsymbol{w}$ and $\boldsymbol{w}'$ such that $\boldsymbol{w} \neq \boldsymbol{w}'$ but $\boldsymbol{x}^T\boldsymbol{w} = 0$ and $\boldsymbol{x}^T\boldsymbol{w}' = 0$ describe the same hyperplane? If yes, what conditions $\boldsymbol{w}$ and $\boldsymbol{w}'$ must meet?

c. For the general case of $\boldsymbol{x} = (x_1, ..., x_N, -1) \in \mathbb{R}^{N+1}$, what is the distance of the hyperplane $\boldsymbol{x}^T\boldsymbol{w} = 0$ from the origin in $\mathbb{R}^N$? Where does the hyperplane intersect with the x_n-axis for $n \in \{1, ..., N\}$?

d. Use the online learning rule you derived in Exercise 3c and describe, in words, how the separating hyperplane in $\mathbb{R}^N$ changes after each update. Make sure you consider the effects of both changing bias/threshold on one side and changing weight parameter $\boldsymbol{w}$ on the other side.

Solution:

a. The line is described by

$$x_2 = -\frac{w_1}{w_2}x_1 + \frac{w_3}{w_2}.$$

hence, its slope is $a = -\frac{w_1}{w_2}$. The line meets the y-axis at $\frac{w_3}{w_2}$ and the x-axis at $\frac{w_3}{w_1}$.

b. Yes, as long as we have

$$\frac{w_1}{w_2} = \frac{w_1'}{w_2'} \quad \text{and} \quad \frac{w_3}{w_2} = \frac{w_3'}{w_2'}.$$

c. The intersect with the x_n-axis is given by

$$x_n = \frac{w_{N+1}}{w_n}.$$

To find the distance, we define $\bar{\boldsymbol{w}} = (w_1, ..., w_N) \in \mathbb{R}^N$ and $\bar{\boldsymbol{x}} = (x_1, ..., x_N) \in \mathbb{R}^N$ and give two different solutions:

Solution 1 (Projection): Consider an arbitrary point $\bar{\boldsymbol{x}}$ on the hyperplane, i.e., $\bar{\boldsymbol{x}}^T\bar{\boldsymbol{w}} = w_{N+1}$. Then, the distance of any point $\bar{\boldsymbol{x}}' \in \mathbb{R}^N$ from the hyperplane is equal to the lenght of the projection of the difference $\bar{\boldsymbol{x}} - \bar{\boldsymbol{x}}'$ on the hyperplane normal vector:

$$d = |\Big(\bar{\boldsymbol{x}} - \bar{\boldsymbol{x}}'\Big)^T \frac{\bar{\boldsymbol{w}}}{||\bar{\boldsymbol{w}}||}|.$$

For the distance to the origin, we need to put $\bar{\boldsymbol{x}}' = 0$:

$$d = |\frac{\bar{\boldsymbol{x}}^T\bar{\boldsymbol{w}}}{||\bar{\boldsymbol{w}}||}| = \frac{|w_{N+1}|}{||\bar{\boldsymbol{w}}||}.$$

Solution 2 (Optimization): The distance of the hyperplane from the origin is given by

$$d = \min_{\bar{\boldsymbol{x}} \text{ s.t. } \bar{\boldsymbol{x}}^T\bar{\boldsymbol{w}} = w_{N+1}} ||\bar{\boldsymbol{x}}||.$$

The solution to this constraint optimization problem should satisfy

$$\frac{\partial}{\partial \bar{\boldsymbol{x}}}\Big(||\bar{\boldsymbol{x}}||^2 - \lambda(\bar{\boldsymbol{x}}^T\bar{\boldsymbol{w}} - w_{N+1})\Big) = 0,$$

where λ is the Lagrange multiplier. This condition combined with the hyperplane constraint, gives us

$$d = ||\frac{w_{N+1}}{||\bar{\boldsymbol{w}}||^2}\bar{\boldsymbol{w}}|| = \frac{|w_{N+1}|}{||\bar{\boldsymbol{w}}||}.$$

d. The updated hyperplane is charachterized by

$$\text{(i)} \quad \bar{\boldsymbol{w}} \leftarrow \bar{\boldsymbol{w}} + \eta\delta(\mu)\bar{\boldsymbol{x}}^\mu$$
$$\text{(ii)} \quad w_{N+1} \leftarrow w_{N+1} - \eta\delta(\mu).$$

Step (i) rotates the hyperplane in a direction to either have it aligned with $\bar{\boldsymbol{x}}^\mu$ (if the target is underestimated, i.e., $\delta(\mu) > 0$) or get aligned with $-\bar{\boldsymbol{x}}^\mu$ (if the target is overestimated, i.e., $\delta(\mu) < 0$). Given the rotated hyperplane, step (ii) moves the hyperplane either down to decrease all the intersect points (if the target is underestimated, i.e., $\delta(\mu) > 0$) or up to decrease all the intersect points (if the target is overestimated, i.e., $\delta(\mu) < 0$).