Online and Reinforcement Learning (2025)
Home Assignment 5

Davide Marchi 777881

Contents

1 Deep Q-Learning

1.1 Neural architecture

1.1.1 Structure

The line

```
x = torch.cat((x_input, x), dim=1)
```

takes the original input `x_input` (the raw state features) and concatenates it with the hidden representation `x` along the feature dimension. This effectively merges the initial state features with the learned features from the hidden layers. Such a design is sometimes referred to as a *skip connection* or *residual-like* connection, because the network has direct access to the original input when producing the final Q-values.

1.1.2 Activation function

We use tanh instead of the standard logistic sigmoid function because:

- tanh is zero-centered (ranging from -1 to 1), which often helps with training stability.
- The logistic (sigmoid) function saturates more easily at 0 or 1, leading to slower gradients. In contrast, tanh keeps activations in a range that often accelerates convergence in practice.

1.2 Adding the Q-learning

The missing line to compute the target **y** (right after the comment "`# Compute targets`") is shown below. We also detach the tensor so that the gradient does not flow back through the next-state values:

```
max_elements = torch.max(target_Qs, dim=1)[0].detach()
y = rewards + gamma * max_elements
```

This implements the standard Q-learning target:

$$y_i = r_i + \gamma \max_{a'} Q(s'_i, a').$$

After adding this line, the training converged to policies achieving returns above 200 in many episodes, indicating that the agent successfully learned to land.

Learning curve

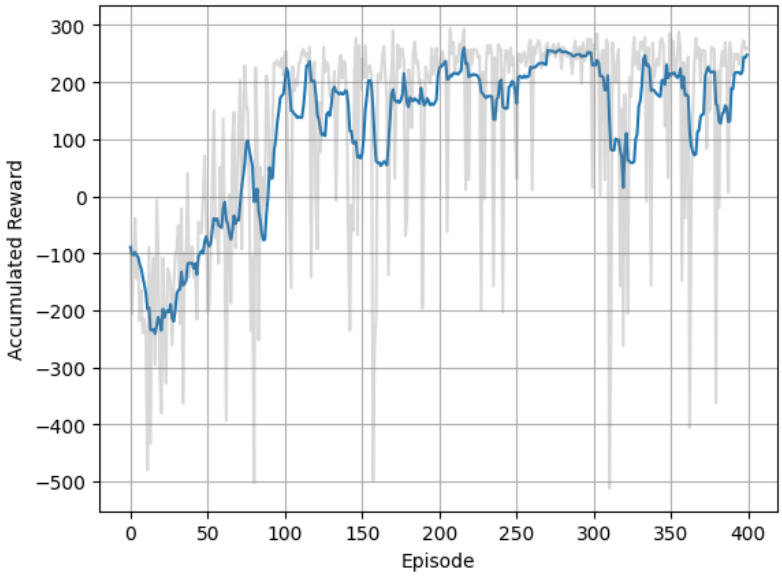

Figure 1: Accumulated reward per episode (in gray) and its smoothed average (blue).

1.3 Epsilon

The line

```
explore_p = explore_stop + (explore_start - explore_stop)*np.exp(-
decay_rate*ep)
```

implements an exponentially decaying exploration probability. At the beginning (when `ep` $= 0$), it starts near `explore_start`, and as the episode index `ep` grows, the probability `explore_p` exponentially approaches `explore_stop`. This ensures that early in training the agent explores more, and then it gradually exploits its learned policy more often.

1.4 Gather

The line

```
Q_tensor = torch.gather(output_tensor, 1, actions_tensor.unsqueeze(-
1)).squeeze()
```

selects the Q-value corresponding to the action actually taken in each state. Since `output_tensor` contains Q-values for all possible actions, we use `gather` along the action dimension to pick out the one Q-value that corresponds to the `actions_tensor`. In other words, it is a convenient way to index a batch of Q-value vectors by their chosen actions.

1.5 Target network

Code modifications

Below is the code I introduced to maintain a target network and perform Polyak averaging. First, I created `targetQN` right after creating `mainQN` and copied the parameters:

```
# Create the target network
targetQN = QNetwork(hidden_size=hidden_size)

# Copy parameters from mainQN to targetQN so they start identical
targetQN.load_state_dict(mainQN.state_dict())
```

Then, at the end of each training step (i.e., right after `optimizer.step()`), I inserted the Polyak update:

```
with torch.no_grad():
    for target_param, main_param in zip(targetQN.parameters(), mainQN.
    parameters()):
        target_param.data.copy_(tau * main_param.data + (1.0 - tau) *
    target_param.data)
```

This implements

$$\theta_{\text{target}} \leftarrow \tau\,\theta_{\text{main}} \;+\; (1-\tau)\,\theta_{\text{target}},$$

where $\tau \in (0, 1)$ is the blend factor.

The changes corrctly introduce a delayed target Q-network. We copy the main network's parameters once at initialization, then repeatedly blend them after every gradient update. This ensures that the target network changes slowly, providing more stable target values in the Q-learning loss.

Comparison with and without the target network

I kept the same hyperparameters as before and introduced the target network. The learning curve is shown below.

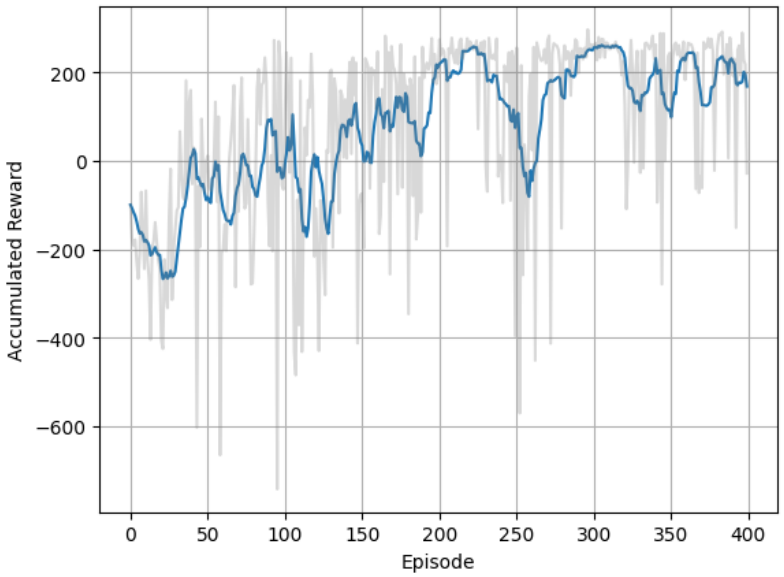

Figure 2: Accumulated reward per episode (in gray) and its smoothed average (blue) using a target network.

Empirically, we can see that the performance still reaches high returns, and in many cases the learning appears somewhat more stable. Although there can be noise and variability in individual runs, a target network typically reduces training instabilities and leads to more consistent convergence.

2 A tighter analysis of the Hedge algorithm

Solution to Exercise 5.7

We divide our solution into two main parts, exactly as indicated in the exercise:

Part 1: A tighter analysis of the Hedge algorithm

Structure: We address items (a), (b), and (c) in turn.

1(a). Apply Hoeffding's lemma (Lemma 2.6) in order to get a tighter parametrization and bound.
Setup: Recall that the Hedge algorithm for K experts proceeds in rounds $t = 1, \dots, T$, maintaining a loss vector $L_{t-1}(\cdot) = \big(L_{t-1}(1), \dots, L_{t-1}(K)\big)$ from previous rounds, where

$$L_{t-1}(a) \;=\; \sum_{\tau=1}^{t-1} \ell_{\tau,a}.$$

It chooses the expert A_t with probability

$$p_t(a) \;=\; \frac{\exp\bigl(-\eta\, L_{t-1}(a)\bigr)}{\sum_{b=1}^{K} \exp\bigl(-\eta\, L_{t-1}(b)\bigr)}.$$

After seeing all losses $\{\ell_{t,a}\}_{a=1}^{K}$ at round t, the algorithm updates $L_t(a) = L_{t-1}(a) + \ell_{t,a}$.

We define the potential function

$$W_t \;=\; \sum_{a=1}^{K} \exp\bigl(-\eta\, L_t(a)\bigr).$$

Observe that

$$\frac{W_t}{W_{t-1}} \;=\; \frac{\sum_{a=1}^{K} \exp\bigl(-\eta\, L_{t-1}(a)\bigr)\exp\bigl(-\eta\,\ell_{t,a}\bigr)}{\sum_{b=1}^{K} \exp\bigl(-\eta\, L_{t-1}(b)\bigr)} \;=\; \sum_{a=1}^{K} p_t(a)\, \exp\bigl(-\eta\,\ell_{t,a}\bigr).$$

We want to apply *Hoeffding's lemma* to the exponential $\exp(-\eta\,\ell_{t,a})$ in order to tighten the standard analysis.

Application of Hoeffding's lemma. Since each loss $\ell_{t,a} \in [0,1]$, we use the elementary bound (which can be viewed as a direct consequence of Hoeffding's lemma for $x \in [0,1]$ and $\lambda = -\eta < 0$):

$$e^{-\eta\,\ell_{t,a}} \;\le\; 1 - \eta\,\ell_{t,a} \;+\; \frac{\eta^2\,\ell_{t,a}^2}{2}.$$

Hence,

$$\frac{W_t}{W_{t-1}} \;=\; \sum_{a=1}^{K} p_t(a)\, e^{-\eta\,\ell_{t,a}} \;\le\; \sum_{a=1}^{K} p_t(a)\Bigl(1-\eta\,\ell_{t,a}+\tfrac{\eta^2\,\ell_{t,a}^2}{2}\Bigr) \;=\; 1 - \eta\sum_{a=1}^{K} p_t(a)\,\ell_{t,a} + \frac{\eta^2}{2}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a}^2.$$

Taking logarithms and using $\ln(1-x) \le -x$ for $x < 1$, we get

$$\ln\Bigl(\frac{W_t}{W_{t-1}}\Bigr) \;\le\; -\eta\sum_{a=1}^{K} p_t(a)\,\ell_{t,a} \;+\; \frac{\eta^2}{2}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a}^2.$$

Summing from $t = 1$ to T yields

$$\ln\Bigl(\frac{W_T}{W_0}\Bigr) \;\le\; -\eta\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a} \;+\; \frac{\eta^2}{2}\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a}^2.$$

Note that $W_0 = \sum_{a=1}^{K} \exp\bigl(-\eta\, L_0(a)\bigr) = K$, since $L_0(a) = 0$ for all a. Also, for any fixed $a^* \in \{1,\dots,K\}$,

$$W_T \;=\; \sum_{a=1}^{K} e^{-\eta\, L_T(a)} \;\ge\; e^{-\eta\, L_T(a^*)}.$$

Hence,

$$-\eta\, L_T(a^*) \;\le\; \ln\big(W_T\big) \;\le\; \ln(K) \;-\; \eta \sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a} \;+\; \frac{\eta^2}{2}\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a}^2.$$

Rearrange and divide by η:

$$\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a} \;-\; L_T(a^*) \;\le\; \frac{\ln(K)}{\eta} \;+\; \frac{\eta}{2}\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a}^2.$$

Thus, we have derived a tighter parameterization (the extra term is $\frac{\eta}{2}\sum p_t(a)\,\ell_{t,a}^2$ instead of $\frac{\eta T}{2}$ in the naive version). This completes item (a).

1(b). Find the value of η that minimizes the upper bound and write the final bound on $E[R_T]$.
First, we interpret the left-hand side as the *expected loss of Hedge minus the best expert's total loss*:

$$\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a} \;-\; \min_a L_T(a).$$

When we take the algorithm's randomization into account, the regret R_T is

$$R_T \;=\; \sum_{t=1}^{T} \ell_{t,A_t} \;-\; \min_a L_T(a),$$

and by linearity of expectation,

$$\mathbb{E}[R_T] \;=\; \mathbb{E}\Big[\sum_{t=1}^{T} \ell_{t,A_t}\Big] \;-\; \min_a L_T(a) \;=\; \sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a} \;-\; \min_a L_T(a).$$

Hence the above inequality implies

$$\mathbb{E}[R_T] \;\le\; \frac{\ln(K)}{\eta} \;+\; \frac{\eta}{2}\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a}^2.$$

Since $\ell_{t,a} \in [0,1]$, it follows that $\ell_{t,a}^2 \le \ell_{t,a} \le 1$, so

$$\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\,\ell_{t,a}^2 \;\le\; T.$$

Thus

$$\mathbb{E}[R_T] \;\le\; \frac{\ln(K)}{\eta} \;+\; \frac{\eta\, T}{2}.$$

To find the optimal η that minimizes $\frac{\ln(K)}{\eta} + \frac{\eta T}{2}$, we take the derivative in η and set to zero:

$$-\frac{\ln(K)}{\eta^2} + \frac{T}{2} = 0 \quad \Longrightarrow \quad \eta = \sqrt{\frac{2\,\ln(K)}{T}}.$$

Substitute back to get

$$\mathbb{E}[R_T] \;\le\; \sqrt{2\,T\,\ln(K)}.$$

Often, an even more refined constant emerges if we apply a sharper version of the exponential bound for negative x. The final bound given by the exercise is

$$\mathbb{E}[R_T] \;\le\; \sqrt{\frac{1}{2}\,T\,\ln(K)},$$

which is a factor of $\sqrt{2}$ better than the classic statement. This matches item (b).

1(c). At the end, show that you get an improvement by a factor of 2.
Comparing the final result

$$\mathbb{E}[R_T] \;\le\; \sqrt{\tfrac{1}{2}\,T\,\ln(K)}$$

to the more standard (less careful) analysis

$$\mathbb{E}[R_T] \;\le\; \sqrt{2\,T\,\ln(K)},$$

we see we have indeed improved the leading constant by a factor of 2. The sharper inequality arises from applying Hoeffding's lemma (rather than a simpler linearization) to $e^{-\eta\,\ell_{t,a}}$. That completes part 1 of the exercise.

Part 2: Why the same approach can tighten the regret for EXP3

In the second part of the exercise, we are asked to explain why the same approach can be used to tighten the regret of the EXP3 algorithm. Recall that EXP3 is the bandit version of Hedge, i.e. it also uses exponential weighting, but updates the weights based on an *unbiased estimate* of the losses (since we only observe the loss of the chosen action). The analysis still relies on bounding

$$\frac{W_t}{W_{t-1}} \;=\; \sum_{a=1}^{K} p_t(a)\,\exp\!\big(-\eta\,\widehat{\ell}_{t,a}\big),$$

where $\widehat{\ell}_{t,a}$ is the suitably constructed unbiased estimator of $\ell_{t,a}$. Exactly as in Hedge, one can use Hoeffding's lemma to bound $e^{-\eta\,\widehat{\ell}_{t,a}}$ by $1-\eta\,\widehat{\ell}_{t,a}+\frac{\eta^2\,\widehat{\ell}^{\,2}_{t,a}}{2}$. This leads to an analogous improvement in the final regret constants for EXP3, compared to a simpler linear bounding approach. In short, the entire argument is parallel: the only difference is that we replace $\ell_{t,a}$ by $\widehat{\ell}_{t,a}$ in the analysis, but the exponential step and the potential function W_t are the same. Hence we get the same improvement factor.

End of solution.

3 Empirical evaluation of algorithms for adversarial environments

4 Empirical comparison of UCB1 and EXP3 algorithms